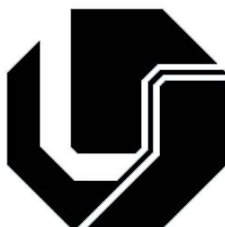


Uma proposta de assinatura digital para imagens por meio de marca d'água

Fernanda Barbosa Coelho Mendes

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo
Programa de Pós-graduação em Ciência da Computação da Universidade Federal de
Uberlândia



Programa de Pós-graduação em Ciência da Computação
Faculdade de Computação
Universidade Federal de Uberlândia
Minas Gerais – Brasil

Fevereiro de 2008

Fernanda Barbosa Coelho Mendes

**UMA PROPOSTA DE ASSINATURA DIGITAL PARA
IMAGENS POR MEIO DE MARCA D'ÁGUA**

Dissertação apresentada ao programa de Pós-graduação em Ciência da Computação da Universidade Federal de Uberlândia, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Área de pesquisa: Redes de Computadores.

Orientador: Prof. Dr. Jamil Salem Barbar

UBERLÂNDIA – Fevereiro de 2008

Dados Internacionais de Catalogação na Publicação (CIP)

- M538p Mendes, Fernanda Barbosa Coelho, 1980-
Uma proposta de assinatura digital para imagens por meio de marca d'água / Fernanda Barbosa Coelho Mendes. - 2007.
134 f. : il.
- Orientador: Jamil Salem Barbar
Dissertação (mestrado) – Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Inclui bibliografia.
1. Redes de computação - Medidas de segurança - Teses. I. Barbar, Jamil Salem. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3-78

Fernanda Barbosa Coelho Mendes

**Uma proposta de assinatura digital para imagens por meio
de marca d'água**

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Uberlândia, para obtenção do título de Mestre.

Área de pesquisa: Redes de Computadores.

Uberlândia, 01 de Fevereiro de 2008

Banca Examinadora

Prof. Dr. Jamil Salem Barbar – UFU (orientador)

Prof. Dr. Walter Godoy Júnior – UTFPR

Prof. Dr. Luís Fernando Faina – UFU

Prof. Dr. Pedro Frosi Rosa – UFU

Aos meus pais Sebastião e Vera Lúcia
e ao meu marido Cláudio,
pelo carinho,
estímulo e compreensão.

“A sabedoria é resplandecente, não murcha,
mostra-se facilmente para aqueles que a amam. Ela se deixa encontrar
por aqueles que a buscam.”

Sabedoria 6,12.

Resumo

Atualmente, os conteúdos digitais podem ser compartilhados e distribuídos na *Internet* com grande facilidade, no entanto muitos intrusos aproveitam-se dessas facilidades para falsificar ou distribuir ilegalmente estes conteúdos. O uso de marca d'água digital é uma solução amplamente adotada para oferecer segurança às transações de arquivos de multimídia digitais por meio da *Internet*. Essa solução é adotada com propósitos de autenticação, proteção de direitos autorais e proteção contra cópia não autorizada. Neste trabalho, é desenvolvido um método para construir uma marca d'água para garantir a assinatura digital de imagens digitais. Além disso, outros requisitos devem ser atendidos por essa marca d'água, tais como a transparência, a verificação pública e o menor complexidade computacional. Para garantir a transparência, a sua inserção é feita por meio do mapeamento das características da imagem hospedeira, utilizando a transformada *wavelet*. Para conseguir menor complexidade computacional, optou-se por utilizar o esquema de *lifting* ao invés de banco de filtros, para implementar a transformada *wavelet*. Para garantir a verificação pública da marca d'água, bem como a sua eficácia contra ataques, optou-se por utilizar as funções *hash* e os algoritmos de assinatura digital que utilizam chaves assimétricas.

Palavras-chave: Redes de computadores, *Internet*, Segurança, Autenticação, Imagens, Marca d'água, *Wavelet*.

Fernanda Barbosa Coelho Mendes

Abstract

Nowadays, digital contents may be shared and easily distributed in Internet, but a lot of intruders use these facilities in order to falsify or illegally distribute this contents. The use of digital watermark is a widely adopted solution, in order to offer security to digital multimedia files by Internet. This solution is adopted with purposes of authentication, copyright and non authorized copy protection. In this dissertation, it is developed a method to construct watermark to assure the authentication and digital sign for digital images. Further, some other requirements may be assisted by this watermark, for example, transparency, public verifying and minimal computer costs. In order to assure transparency, the insertion is made through mapping host image characteristics, by use of wavelet. In order to get a minimal computer costs, it was decided to use lifting scheme instead of filter bank, to implement wavelet transform. To assure watermark public verifying as well security against attacks, it was used hash functions and cryptographic algorithms that use non asymmetric or public keys.

Keywords: Network, Internet, Security, Authentication, Images, Watermark, Wavelet.

Copyright

Copyright © 2008 by FERNANDA BARBOSA COELHO.

“The copyright of this thesis rests with the author. No quotations from it should be published without the author’s prior written consent and information derived from it should be acknowledged”.

Agradecimentos

Primeiramente a Deus, que me deu esta oportunidade, caminhou ao meu lado durante todo o percurso e me rodeou de pessoas maravilhosas que muito me apoiaram.

À minha mãe, Vera Lúcia, por suas orações, seu encorajamento, seu companheirismo e por madrugar junto comigo.

À meu pai Sebastião, por seu exemplo de perseverança.

Ao meu marido Dim, por seu imensurável apoio, carinho, compreensão e por sempre me haver esperado chegar de viagem, tarde da noite, junto com meus pais e irmãos.

Ao meu orientador, Jamil, por seu empenho em me ajudar, sempre.

Aos meus amigos, Gustavo e Grings, pelo imensurável apoio, pela força e pelo incentivo.

Aos meus amigos do mestrado, Italo, Ricardo, Pedro e Lúcio pelo apoio e companheirismo.

Aos professores e funcionários do mestrado e da FACOM.

À FACOM e ao Programa de Pós-graduação em Ciência da Computação pelo apoio e oportunidade.

À CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, pelo apoio financeiro concedido.

Sumário

Lista de Figuras	xiii
Lista de Tabelas	xv
Lista de Símbolos	xvi
Lista de Acrônimos	xviii
1 Introdução	1
2 O Uso de Transformadas na Análise das Características de Imagens Digitais	7
2.1 Métodos para Representação e Reconstrução de Sinais Digitais	8
2.1.1 Representação por Bases	9
2.1.2 Representação por <i>Frames</i>	11
2.1.3 Representação em Multi-resolução	11
2.2 A Análise de Funções para a Determinação de suas Características	13
2.2.1 O Espectro de Freqüência de uma Função e a Exibição de suas Características	14
2.3 A Transformada de <i>Fourier</i>	15
2.3.1 A Transformada de <i>Fourier</i> no Espaço $L^1(\mathbb{R})$	16
2.3.2 A Transformada de <i>Fourier</i> no Espaço $L^2(\mathbb{R})$	18
2.3.3 O Uso da Transformada de <i>Fourier</i> na Representação de Funções . .	19
2.3.4 A Análise de Freqüência e a Eficácia da Transformada de <i>Fourier</i> .	19
2.3.5 A Transformada Janelada de Fourier	20
2.4 A Transformada <i>Wavelet</i>	23
2.4.1 O Que É <i>Wavelet</i> ?	23

2.4.2	Famílias de Funções <i>Wavelets</i>	24
2.4.3	A Transformada <i>Wavelet</i> Inversa	27
2.4.4	A Transformada <i>Wavelet</i> Discreta	28
2.4.5	A Transformada <i>Wavelet</i> Discreta e a Análise em Multi-resolução	30
2.5	Comparação Entre a Transformada <i>Wavelet</i> e a de <i>Fourier</i>	32
2.6	O Uso da Transformada <i>Wavelet</i> na Inserção de Marca D'água Digital	33
3	Integridade e autenticidade de imagens digitais	38
3.1	Técnicas de Autenticação e Assinatura Digital	39
3.2	Códigos Autenticadores	41
3.3	O Uso de Mensagens Cifradas como Código Autenticador	42
3.4	O Uso de MAC como Código Autenticador	43
3.5	O Uso de Código <i>Hash</i> Como Autenticador	44
3.5.1	A Função <i>Hash</i> RIPEMD-160	46
3.6	Criptografia Assimétrica e as Técnicas de Assinatura Digital	47
3.6.1	A Assinatura Digital RSA	49
3.6.2	O Uso de Assinatura Digital para Autenticação de Imagens	50
3.7	Marca D'água Digital	51
3.7.1	Ataques Contra Marca D'Água para Autenticação	53
3.7.2	Características de Marcas D'água para Autenticação	56
3.7.3	Trabalhos Relacionados	58
3.7.4	Aspectos de Segurança de Marca D'água Digital	62
4	Uma Proposta de Assinatura Digital para Imagens por Meio de Marca D'água	64
4.1	Ambiente de Desenvolvimento	66
4.2	Implementação da DWT e da IDWT Utilizando o Esquema de <i>Lifting</i>	66
4.2.1	A Implementação da DWT	67
4.2.2	A Implementação da IDWT	74
4.3	Construção da Marca D'água	78
4.4	Inserção da Marca D'água nos Coeficientes <i>Wavelet</i>	84
4.5	Extração e Verificação da Marca D'água	89
4.5.1	Verificação da Assinatura Digital	92
4.5.2	Localização de Adulterações	94

4.6	Experimentos e Resultados Obtidos	95
4.6.1	Qualidade da Imagem Marcada e <i>Performance</i> Computacional . . .	96
4.6.2	Aspectos de Segurança, Detecção e Localização de Adulterações . .	98
4.6.3	Propriedades da Proposta de Assinatura Digital para Imagens De- senvolvida	104
5	Conclusões	106
	Referências Bibliográficas	110
	Apêndice A	115
	Anexo A	130

Lista de Figuras

1.1	Ataques na rede <i>Internet</i>	2
1.2	Classificação de marca d'água digital	4
2.1	O operador de representação por <i>frames</i>	12
2.2	Gráficos da função seno com frequência 4, 8 e 16.	14
2.3	Efeito da função de modulação.	21
2.4	Gráfico de uma <i>wavelet</i>	24
2.5	Algumas <i>wavelets</i>	24
2.6	Escalas de uma função.	25
2.7	Dilatação da <i>wavelet</i> chapéu mexicano.	26
2.8	Translação da <i>wavelet</i> chapéu mexicano.	26
2.9	Similaridade entre uma <i>wavelet</i> e uma função.	27
2.10	Representação em multi-resolução decomposição e reconstrução do sinal c_n	32
2.11	Gráfico da função <i>wavelet</i> CDF 9/7 bi-ortogonal	34
2.12	Decomposição em dois níveis de um sinal ao aplicar a transformada <i>wavelet</i> bidimensional	36
2.13	Decomposição de uma imagem por meio da DWT	37
3.1	Ataque do homem do meio: Malice interfere na comunicação entre Alice (servidor) e Bob (cliente) para forjar a marca d'água	54
4.1	Visão geral da proposta apresentada.	65
4.2	Diagrama de blocos dos três passos do esquema <i>lifting</i>	67
4.3	Hierarquia de sub-bandas resultante da decomposição <i>wavelet</i> nível 3.	72
4.4	Diagrama de blocos dos três passos do esquema <i>lifting</i> que implementa a IDWT	75
4.5	Processo de construção da marca d'água M	79

4.6	Decomposição da imagem Lenna.	80
4.7	Imagens que constituem a marca d'água.	84
4.8	Processo de inserção da marca d'água M na imagem I	85
4.9	Processo de extração da marca d'água M da imagem I	90
4.10	Processo de verificação da assinatura digital	93
4.11	Invisibilidade da marca d'água.	96
4.12	Fragilidade da marca d'água e capacidade de detectar as regiões adulteradas.	99
4.13	Detecção da alteração no tamanho da imagem Lenna.	100
4.14	Fragilidade da marca d'água a compressão.	101
4.15	Detecção do ataque vetor de quantização	102
4.16	Detecção do ataque à marca d'água.	103
1	Figuras utilizadas nos testes.	130
2	Figuras utilizadas nos testes (continuação).	131
3	Figuras utilizadas nos testes (continuação).	132
4	Figuras utilizadas nos testes (continuação).	133
5	Figuras utilizadas nos testes (continuação).	134
6	Figuras utilizadas nos testes (continuação).	134

Lista de Tabelas

3.1	O uso básico de mensagens cifradas	42
3.2	O uso de MAC para autenticação de mensagens	43
3.3	O uso de função <i>hash</i> para autenticação de mensagens	45
3.4	<i>Performance</i> relativa das funções <i>hash</i> MD5, SHA-1 e RIPEMD-160	47
4.1	Resultados	97

Lista de Símbolos

Letras latinas

a_j - Coeficiente que mede a amplitude de um componente de frequência.

B - Base de um espaço de funções.

C_ψ - Constante de admissibilidade da função ψ .

(c_j) - Seqüência numérica de números reais ou complexos.

E - Energia de uma função.

(e_j) - Vetor pertencente à base B .

$\mathcal{F}$ - Espaço de funções.

F - Operador de representação do espaço de funções $\mathcal{F}$.

$\hat{f}$ - Transformada de *Fourier* da função f .

$\tilde{f}(s, t)$ - Transformada *wavelet* contínua da função f .

$\hat{f}^{-1}$ - Inversa da transformada de *Fourier*.

$f(t)$ - Função ou sinal representado no domínio do tempo ou do espaço.

$\mathcal{H}$ - Espaço de *Hilbert*.

i, j, m, n - Números inteiros utilizados como índice.

$L^1(\mathbb{R})$ - Espaço das funções integráveis.

$L^2(\mathbb{R})$ - Espaço das funções de quadrado integrável.

$L_T^2(\mathbb{R})$ - Espaço das funções de quadrado integrável cujo período é T .

l^2 - Espaço de seqüências de quadrado somáveis que é uma representação de $L^2(\mathbb{R})$.

- R^{-1} - Operador inverso do operador de representação $R(f)$.
- $R(f)$ - Operador de representação de funções.
- $\mathcal{S}$ - Espaço de seqüências numéricas.
- $S_{m,n}$ - Coeficientes *wavelet* de aproximação discretos.
- s - Valor contínuo que determina a dilatação da função *wavelet* mãe.
- s_m - Valor discreto determina a dilatação da função *wavelet* mãe.
- $\{V_j\}$ - Sub-espaço fechado pertencente a uma seqüência de sub-espaços.
- T - Período de uma função.
- $T_{m,n}$ - Coeficientes *wavelet* de detalhe discretos.
- t - Valor contínuo que determina a translação da função *wavelet* mãe.
- $t_{m,n}$ - Valor discreto que determina a translação da função *wavelet* mãe.

Letras gregas

- (α_j) - Seqüência numérica.
- $\{\varphi_n\}, \{\varphi_m\}$ - Coleção de funções de um espaço de *Hilbert* ou *frame*.
- $\tilde{\varphi}_j$ - *Frame* recíproco do *frame* φ_j .
- $\phi_{m,n}$ - Função *wavelet* pai.
- $\hat{\psi}(f)$ - Transformada de *Fourier* da função ψ .
- $\psi_{m,n}$ - Função *wavelet* mãe da transformada discreta.
- $\psi(t)$ - Função *wavelet* mãe da transformada contínua.
- ω_j - Valor de freqüência.

Lista de Acrônimos

ACK - *Acknowledgement*

CDF - *Cohen, Daubechies e Feauveau*

DCT - *Discrete Cosine Transform*

DES - *Data Encryption Standard*

DFT - *Discrete Fourier Transform*

DSA - *Digital Signature Algorithm*

DWT - *Discrete Wavelet Transform*

FBI - *Federal Bureau of Investigation*

FCS - *Frame Check Sequence*

FIR - *Finite Impulse Response*

FPA - *Filtro Passa Alta*

FPB - *Filtro Passa Baixa*

mdc - máximo divisor comum

HMAC - *Hash Message Authentication Code*

HVS - *Human Visual System*

IDWT - *Inverse Discrete Wavelet Transform*

IP - *Internet Protocol*

IPSec - *IP Security Protocol*

JND - *Just Noticeable Distortion*

JPEG - *Joint Photographic Experts Group*

KB - *Kilo Bytes*

LSB - *Least Significant Bits*

LUT - *look-up table*

MAC - *Message Authentication Code*

MB - *Mega Bytes*

MD4 - *Message Digest Algorithm 4*

MD5 - *Message Digest Algorithm 5*

MHz - *Mega Hertz*

MIME - *Multipurpose Internet Mail Extention*

MSB - *Most Significant Bits*

MSE - *Mean Squared Error*

NACK - *Negative Acknowledgement*

PDF - *Formato de Documento Portável*

PGP - *Pretty Good Privacy*

PSNR - *Peak Signal-to-Noise Ratio*

RAM - *Randomic Access Memory*

RIPEMD - *Race Integrity Primitives Evaluation Message Digest*

RSA - *Rivestn Shamir Adleman*

SHA - *Secure Hash Algorithm*

SHA-1 - *Secure Hash Algorithm - Version 1*

SET - *Secure Eletronic Transaction*

SSL - *Secure Sockets Layer*

STFT- *Short Time Fourier Transform*

S/MIME - *Secure MIME*

TCP - *Transmission Control Protocol*

TLS - *Transport Layer Security*

VPN - *Virtual Private Network*

WFT - *Windowed Fourier Transform*

Capítulo 1

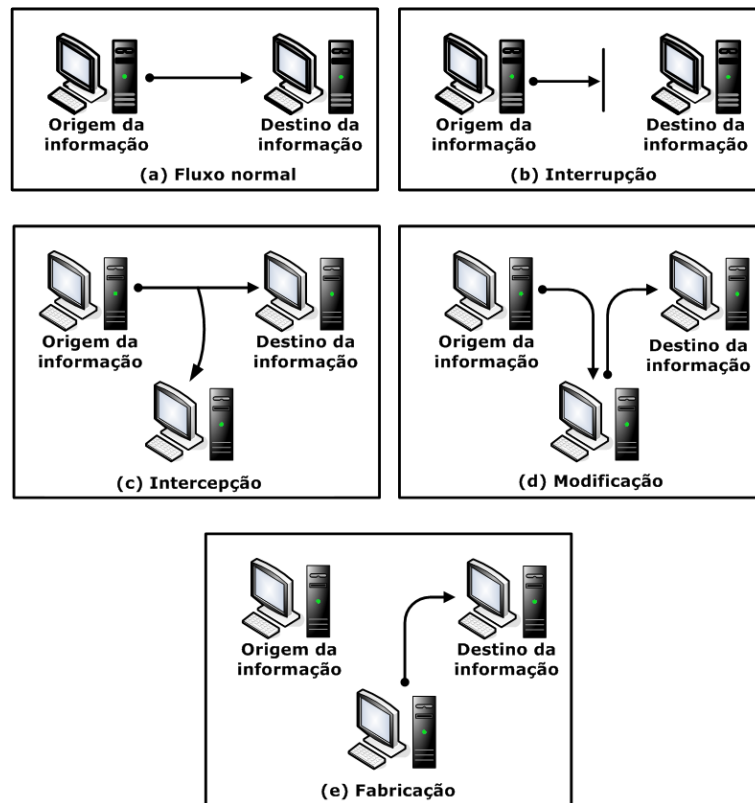
Introdução

Nas últimas décadas, a *Internet* tornou-se uma das mais importantes fontes de informação e passou a oferecer um canal de comunicação para tráfego de todos os tipos de dados. Com a disseminação do uso de conteúdos digitais, tem sido amplamente utilizada para distribuição, comercialização, armazenamento e transmissão de conteúdos digitais tais como imagens, áudio e vídeo.

No cenário atual, os arquivos de multimídia são muito utilizados em aplicações computacionais e podem ser compartilhados e distribuídos em sítios na *Internet* com grande facilidade. Por exemplo, algumas aplicações médicas disponibilizam imagens provenientes de equipamentos, como as tomografias computadorizadas, ressonâncias magnéticas e ultrassonografias, para serem processadas em ambiente distribuído. Há, também, uma variedade de sítios, como os de empresas de publicidade, de compra e venda de mercadorias e de acervos digitais, que disponibilizam os seus arquivos de multimídia com diversas finalidades, tais como publicação de notícias, exibição de mercadorias ou obras artísticas, dentre outras.

No entanto, na rede *Internet*, os canais de comunicação são abertos e inseguros, de forma que as informações que neles trafegam ficam sujeitas a diversos ataques, como interrupção, interceptação, modificação ou fabricação de informações falsificadas, por parte de entidades não autorizadas. A Figura 1.1 ilustra como é cada uma destas categorias de ataques.

Além da insegurança dos canais de comunicação, atualmente, existem diversas ferramentas específicas para manipulação de arquivos de multimídia digitais. Assim, muitos intrusos se aproveitam destas ferramentas e da insegurança da rede *Internet* e conseguem, facilmente, fazer cópias ilegais ou alterar o conteúdo destes arquivos. O problema é que,

Figura 1.1: Ataques na rede *Internet*

normalmente, há um valor, seja ele de ordem monetária ou não, agregado aos arquivos de multimídia disponibilizados. Portanto, é fundamental garantir a sua integridade e autenticidade.

Para oferecer segurança a certas aplicações e camadas de protocolos, diversos mecanismos foram desenvolvidos para atuarem nas diferentes camadas da pilha de protocolos da *Internet*. Dentre esses mecanismos, tem-se o protocolo IPSec (*IP Security Protocol*) [1], que é implementado na camada IP (*Internet Protocol*). Por meio de seu intermédio, uma VPN (*Virtual Private Network*) é estabelecida entre as partes comunicantes, para garantir a transmissão segura de informações.

Além desses, os protocolos PGP (*Pretty Good Privacy*) [2] e S/MIME (*Secure/Multipurpose Internet Mail Extension*), são implementados na camada de aplicação, para oferecer os serviços de confidencialidade e autenticação, podendo ser utilizados nas aplicações de correio eletrônico e de armazenamento de arquivos; o serviço de autenticação *Kerberos*, para aplicações do tipo cliente/servidor; o conjunto de protocolos SET (*Secure Electronic Transaction*), desenvolvido para oferecer segurança às transações de cartão de crédito por meio da *Internet*.

Há também os protocolos SSL (*Secure Sockets Layer*) e TLS (*Transport Layer Security*), desenvolvidos para oferecer segurança às aplicações *Web*. Esses mecanismos utilizam diversas ferramentas criptográficas para autenticar, cifrar e decifrar mensagens, como o RSA (*Rivest-Shamir-Adleman*), o HMAC (*Hash Message Authentication Code*), o MD5 (*Message Digest Algorithm 5*) e o SHA-1 (*Secure Hash Algorithm - Version 1*).

Todavia, nenhum desses mecanismos lida especificamente com o problema da segurança dos arquivos de multimídia digitais disponibilizados na *Internet*. A marca d'água digital é uma solução para coibir a falsificação e distribuição ilegal destes arquivos. Alguns tipos de marca d'água são capazes não só de detectar, mas também de localizar alterações numa imagem marcada com uma resolução previamente estabelecida.

Uma marca d'água é um sinal portador de informação acrescentado ao conteúdo digital que, posteriormente, pode ser extraído e analisado para que se faça alguma alegação sobre esse conteúdo [3], sendo classificada em robusta, frágil e semi-frágil. Essa classificação geralmente determina a finalidade para a qual a marca d'água é utilizada [4].

A marca d'água robusta é projetada para resistir mesmo quando o conteúdo digital sofre alterações bruscas, por isto é utilizada para proteção de direitos autorais. Ao contrário da robusta, a marca d'água frágil é extremamente sensível a qualquer modificação no conteúdo digital, sendo facilmente corrompida por qualquer processamento. Essa característica torna-a útil para verificação de autenticidade. A marca d'água semi-frágil é caracterizada como um meio-termo entre a robusta e a frágil, pois resiste a algumas alterações suaves, que não modificam o conteúdo visual, enquanto é sensível às alterações radicais como a edição. Por isto, esse tipo de marca d'água também é utilizado para autenticação, porém permite, por exemplo, que a imagem marcada seja compactada antes de ser transmitida na *Internet* [3].

Atualmente, esses três tipos de marca d'água digital são amplamente adotados com propósitos de autenticação, proteção de direitos autorais e proteção contra cópia não autorizada. Porém, para atingir efetivamente esses propósitos, a marca d'água deve satisfazer uma série de requisitos.

Além dessa classificação com relação à robustez, as marcas d'água também são classificadas como *blind*, quando o processo de extração não requer a imagem original ou *non-blind*, quando a marca d'água não pode ser extraída sem o uso da imagem original. Outra classificação refere-se ao tipo de chave utilizada no método de extração e verificação. Nesse caso podem ser públicas ou privadas. A Figura 1.2 ilustra as três formas

de classificação de marca d'água.

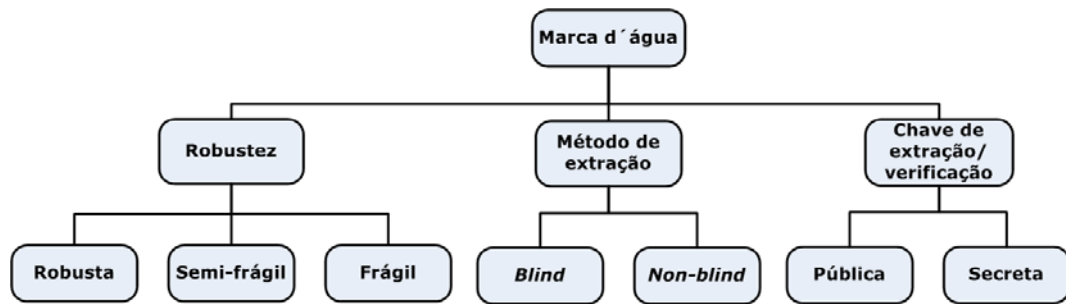


Figura 1.2: Classificação de marca d'água digital

A possibilidade de detecção e verificação pública da marca d'água é um requisito, há pouco tempo, raramente proposto em esquemas de marca d'água digital. Esse requisito sugere uma associação entre marca d'água e algoritmos criptográficos assimétricos, mais precisamente, assinaturas digitais organizadas adequadamente para este fim.

Há várias maneiras de inserir a marca d'água em imagens digitais. Entretanto, é mais razoável inseri-la nas partes da imagem onde o sinal adicionado não provoque distorções ou alterações visuais bruscas, mantendo a compatibilidade com o HVS (*Human Visual System* - Sistema Visual Humano) [5]. O uso da transformada *wavelet* possibilita a identificação dessas partes, pois permite uma análise estatística das características de um sinal, neste caso, uma imagem digital. Por isto, essa transformada é amplamente adotada, não só em esquemas de marca d'água, mas também em métodos de compressão de imagens e em diversas áreas do processamento digital de sinais.

A escolha da função *wavelet* apropriada é extremamente importante para que, ao aplicar a transformada, obtenha-se uma melhor aproximação ou representação do sinal original [6]. Além disto, o uso da função apropriada também é importante para que seja possível localizar, no sinal transformado, as regiões que possuem as características estabelecidas (de acordo com a finalidade da aplicação).

A *wavelet* CDF (Cohen, Daubechies e Feauveau) 9/7 bi-ortogonal, pertencente à família Cohen Daubechies Feauveau, é amplamente adotada no processamento digital de imagens, por causa de suas propriedades, que garantem excelente localização no domínio tempo-frequência¹, e da sua boa compatibilidade com o HVS (Sistema Visual Humano). Por causa dessas características, essa *wavelet* é utilizada pelo FBI (*Federal Bureau of*

¹A boa localização no domínio tempo-frequência é uma característica extremamente importante das ferramentas para análise de sinais, pois possibilita não só a detecção, mas também a localização das características do sinal.

Investigation) para compressão de imagens de impressões digitais e, juntamente com a *wavelet* LeGall 5/3, tornou-se a tecnologia básica nos padrões JPEG2000².

Existem dois métodos para construir a transformada *wavelet*: o método clássico, via banco de filtros, e o esquema de *lifting*. A principal diferença entre esses métodos é que o esquema de *lifting* não utiliza a transformada de *Fourier* em sua implementação. De acordo com Daubechies, qualquer transformada *wavelet* discreta ou filtros do tipo FIR (*Finite Impulse Response*) podem ser decompostos em uma seqüência simples de passos de *lifting* [7].

A implementação da transformada *wavelet* discreta por meio do esquema de *lifting* é mais simples e rápida do que aquela via banco de filtros, pois reduz a complexidade computacional pela metade. Além disso, o esquema de *lifting* reduz o custo em termos de memória, pois possibilita o seu uso eficiente por meio da substituição de variáveis [7]. Portanto, o resultado é uma implementação computacionalmente menos complexa e onerosa. Outra vantagem é que esse esquema possibilita a definição da transformada *wavelet*, que mapeia inteiros em inteiros, cujos coeficientes são valores inteiros [7].

A transformada *wavelet* de coeficientes inteiros, denominada transformada *wavelet* inteira, é apropriada para aplicações que requerem processamento de sinal sem perda de informações e é compatível com a maioria das aplicações para processamento de imagens, que normalmente trabalham com valores inteiros [8].

Por isso, essa transformada é utilizada nos processos de decomposição e reconstrução de imagens do método descrito nesta dissertação. Por meio dessa transformada, a imagem original é decomposta em coeficientes *wavelets*, que são organizados em uma hierarquia de sub-bandas de frequência. Dentre essas sub-bandas, a de frequência mais baixa possui as características que garantem que a inserção de uma marca d'água de robustez razoável não provoque distorções perceptíveis na imagem. A marca d'água é inserida nos coeficientes dessa sub-banda de frequência. Após a inserção, a transformada *wavelet* inversa é aplicada nos coeficientes e a imagem marcada é obtida.

Esta dissertação apresenta um método de autenticação de imagens digitais por meio de marca d'água frágil. Na elaboração deste método, são exploradas as propriedades da *wavelet* CDF 9/7 e da transformada *wavelet* inteira, as vantagens da implementação dessa transformada por meio do esquema *lifting* e a segurança dos recursos criptográficos, tais como função *hash* e chaves simétricas.

²JPEG 2000 é um conjunto de padrões criado pelo comitê JPEG (*Joint Photographic Experts Group*).

A proposta de assinatura digital para imagens desenvolvida neste trabalho tem como diferencial o algoritmo de construção da marca d'água, que utiliza criptografia de chave pública e função *hash* para criar uma assinatura digital da imagem hospedeira. Essa assinatura digital, juntamente com uma imagem binária significativa, faz parte da constituição da marca d'água digital que tem como funcionalidade a localização das regiões adulteradas.

A finalidade desse método é garantir que a integridade e autenticidade das imagens digitais disponibilizadas para acesso via *Internet* possam ser verificadas por meio de uma marca d'água segura³, publicamente verificável, compatível com o HVS e capaz de localizar adulterações numa imagem marcada. Além disso, menor complexidade computacional também é um requisito importante que o método deve atender.

Esta dissertação, além deste capítulo introdutório, está organizada em outros quatro capítulos, descritos resumidamente a seguir.

O Capítulo 2 descreve o problema da representação de funções e as ferramentas matemáticas que o resolvem, tais como as séries de *Fourier*, a transformada de *Fourier*, a transformada janelada de *Fourier* e a transformada *wavelet*. Essas transformadas são comparadas e as vantagens de seu uso em processamento digital de sinais analisadas.

O Capítulo 3 aborda o problema da autenticação de imagens digitais. São apresentados os possíveis ataques a imagens digitais, o conceito e os tipos de aplicações de marca d'água digital e os conceitos de autenticação e de assinatura digital. Também são apresentadas as ferramentas criptográficas capazes de garantir autenticação e assinatura digital e como tais ferramentas podem ser aplicadas juntamente com a marca d'água para oferecer segurança a imagens.

O Capítulo 4 apresenta o método de autenticação de imagens desenvolvido: os algoritmos de construção, inserção, extração e verificação da marca d'água e os resultados obtidos com vários testes que visam a validar o método.

O Capítulo 5 encerra o trabalho, com as considerações finais, incluindo o levantamento da contribuição deste trabalho à comunidade científica e as propostas para trabalhos futuros.

O Apêndice A apresenta o código-fonte do aplicativo para autenticação de imagens implementado no decorrer do desenvolvimento desta dissertação.

³Este termo, nesta dissertação, é utilizado para designar a eficácia da marca d'água contra ataques.

Capítulo 2

O Uso de Transformadas na Análise das Características de Imagens Digitais

Analisar e interpretar sinais por meio dos sentidos faz parte da rotina diária da maioria dos seres vivos. Os órgãos capturam e analisam o sinal para gerarem uma representação que é enviada ao cérebro que, por sua vez, faz a interpretação do sinal recebido. É por meio desse processo que se percebem sinais como cores, sons e imagens.

Os sinais são caracterizados por funções: a cada instante no decorrer do tempo, para cada ponto no espaço, a função que o caracteriza produz uma certa saída que pode ser detectada por órgãos do sentido ou sensores artificiais. Com a proliferação e crescimento contínuo do poder computacional, a capacidade de analisar e interpretar sinais deixa de ser uma característica exclusivamente dos seres vivos e o interesse nos benefícios do processamento digital de sinais torna-se cada vez maior.

Atualmente, muitos sinais podem ser tratados por processadores e armazenados em mídias digitais. Além disso, há uma proliferação do uso da *Internet* como meio de distribuição, comercialização e transmissão de conteúdos digitais tais como imagens, áudio e vídeo. A marca d'água é uma solução utilizada para oferecer proteção a esses arquivos com relação a determinados aspectos tais como reconhecimento de propriedade, proteção contra cópia ilegal e garantia de integridade e de autenticidade. Quando se trata de arquivos de multimídia, em particular de imagens, a invisibilidade é um requisito importante que a marca d'água deve atender.

Para atender esse requisito, é necessário utilizar ferramentas que possibilitem analisar as características da imagem digital, determinando em quais de suas regiões em que a inserção da marca d'água provoca o mínimo de distorções.

O uso de ferramentas matemáticas, tais como a transformada *wavelet* e a transformada de *Fourier*, é fundamental na análise de sinais digitais, particularmente de imagens, pois, por meio delas, pode-se gerar representações ou aproximações desses sinais que possibilitem extrair, ou determinar as suas características principais.

Este capítulo apresenta os conceitos relacionados com a análise de imagens digitais por meio de transformadas: os aspectos teóricos e as diferenças das duas transformadas mais populares, a de *wavelet* e a de *Fourier*.

2.1 Métodos para Representação e Reconstrução de Sinais Digitais

Esta seção descreve o que é a representação de sinais e apresenta quais são os principais métodos de representação. Esta e as próximas seções apresentam os aspectos teóricos, lidam com conceitos matemáticos genéricos, por isso, o termo função representa o sinal a ser analisado.

Seja $\mathcal{S}$ um espaço de seqüências numéricas $(c_j)_{j \in \mathbb{Z}}$, de números reais ou complexos, tal que $\mathcal{S}$ possui uma norma¹ que permite calcular a distância entre seqüências do espaço. A representação de um espaço de funções $\mathcal{F}$ é um operador $F : \mathcal{F} \rightarrow \mathcal{S}$. Para uma dada função f , sua representação $R(f)$ é uma seqüência:

$$R(f) = (f_j)_{j \in \mathbb{Z}} \in \mathcal{S}, \quad (2.1)$$

onde R é denominado o operador de representação. Se R é linear e contínuo, tem-se uma representação linear.

Os exemplos mais importantes de representação ocorrem quando as funções pertencem ao espaço das funções de quadrado integrável, denominado $L^2(\mathbb{R})$, definido por [9]:

$$L^2(\mathbb{R}) = \{f : \mathbb{R} \rightarrow \mathbb{R}, \int_{\mathbb{R}} |f(t)|^2 dt < \infty\}, \quad (2.2)$$

¹Em matemática, norma é uma função que associa cada vetor de um espaço em um número real não negativo. O conceito de norma é estreitamente ligado ao comprimento de um vetor.

e a representação do espaço $L^2(\mathbb{R})$ é o espaço l^2 seqüências de quadrado somáveis,

$$l^2 = \left\{ (x_i)_{i \in \mathbb{Z}}; x_i \in \mathbb{C}; \sum_{i=-\infty}^{+\infty} |x_i|^2 < \infty \right\}. \quad (2.3)$$

Quando o operador de representação é inversível, pode-se reconstruir uma função f a partir de sua representação: $f = R^{-1}((f_i)_{i \in \mathbb{Z}})$. Nesse caso, tem-se uma representação perfeita ou representação ideal.

Quando determinada função é implementada utilizando uma linguagem de programação, constrói-se, na verdade, uma representação para essa função. Existem vários métodos de se calcular a representação e a reconstrução de funções, como: por bases, por *frames*, por multi-resolução, por bases de Riesz, por projeção, *Galerkin*, dentre outros [9]. Dentre esses métodos, os três primeiros são os de maior relevância para esta dissertação, porque fornecem embasamento teórico para o entendimento da transformada *wavelet*. Esses métodos estão descritos sucintamente nas próximas seções.

2.1.1 Representação por Bases

A idéia básica de representar funções por bases consiste em decompô-las utilizando um conjunto finito de funções mais simples, ou seja, construir bases de um espaço.

Um conjunto $B = \{e_j; j \in \mathbb{Z}\}$ é uma base do espaço de funções $\mathcal{F}$ se os vetores e_j forem linearmente independentes² e, para cada $f \in \mathcal{F}$ existir uma seqüência $(\alpha_j)_{j \in \mathbb{Z}}$ de números, tais que [9]:

$$f = \sum_{j \in \mathbb{Z}} \alpha_j e_j \quad (2.4)$$

Na representação por base, o operador de representação é definido como:

$$R(f) = (\alpha_j)_{j \in \mathbb{Z}}. \quad (2.5)$$

A Equação 2.4 reconstrói a função f a partir da seqüência de representação α_j .

Os espaços de *Hilbert* são um caso particular de grande importância que ocorrem quando o espaço vetorial é completo³ e possui norma e produto interno [6]. Esses espaços,

²Um vetor é considerado linearmente independente quando ele, como unidade básica, não apresentar relação linear com outros vetores que também são unidades básicas de um espaço vetorial. Para que um vetor seja considerado linearmente independente em relação a outro vetor, qualquer tentativa de combinação linear entre eles deve gerar um vetor nulo.

³Um espaço vetorial é completo se todas as seqüências de Cauchy convergem para um limite no espaço.

que despertam grande interesse, possuem algumas características especiais, descritas na próxima subseção.

Conjunto Ortonormal Completo

Uma coleção de funções $\{\varphi_n; n \in \mathbb{Z}\}$ de um espaço de *Hilbert* $\mathcal{H}$ é um *conjunto ortonormal completo* se as seguintes condições forem satisfeitas [9]:

1. **Ortogonalidade:** $\langle \varphi_m, \varphi_n \rangle = 0$, se $m \neq n$;
2. **Normalização:** $\|\varphi_n\| = 1 \quad \forall n \in \mathbb{Z}$;
3. **Completeness:** para todo $f \in \mathcal{H}$, $N > 0$, e para qualquer $\varepsilon > 0$,

$$\left\| f - \sum_{k=-N}^N \langle f, \varphi_k \rangle \varphi_k \right\| < \varepsilon. \quad (2.6)$$

A terceira condição diz que combinações lineares de funções do conjunto podem ser utilizadas para gerarem aproximações de funções arbitrárias do espaço. Por isso, conjuntos completos ortonormais são também conhecidos como *bases ortonormais* dos espaços de *Hilbert* [9].

Uma base ortonormal $\{\varphi_j\}$ define um operador de representação, $R : \mathcal{H} \rightarrow l^2$, $R(f) = (f_j) = (\langle f, \{\varphi_j\} \rangle)$, inversível. Portanto, a representação é exata. A reconstrução do sinal original é dada por [9]:

$$f = \sum_{k=-\infty}^{+\infty} \langle f, \varphi_k \rangle \varphi_k. \quad (2.7)$$

Pode-se observar que a condição de ortogonalidade implica que os elementos φ_n são linearmente independentes, o que implica que a seqüência de representação $(\langle f, \{\varphi_j\} \rangle)$ é determinada unicamente por f . Além disso, essa representação preserva a norma, que é dada pela expressão conhecida como *equação de Plancherel*:

$$\|R(f)\|^2 = \sum_{k \in \mathbb{Z}} \langle f, \varphi_k \rangle^2 = \|f\|^2. \quad (2.8)$$

Em outras palavras, que definem de forma mais resumida e prática, uma base ortonormal é um conjunto de vetores, perpendiculares entre si, cuja combinação pode definir completamente uma função f [6].

2.1.2 Representação por *Frames*

A representação por *frames* visa a sanar a dificuldade de se construírem conjuntos ortonormais completos, devido às restrições que esse tipo de conjunto deve atender. Com esse intuito, utilizam-se coleções de funções que não constituem, necessariamente, um conjunto ortonormal, mas que podem ser utilizadas para definir um operador de representação. Tais coleções são denominadas *frames*.

Uma coleção de funções $\{\varphi_n \mid n \in \mathbb{Z}\}$ é um *frame* se existirem duas constantes A e B tais que $0 < A \leq B < +\infty$, tal que para todo $f \in \mathcal{H}$ tem-se a seguinte restrição satisfeita:

$$A\|f\|^2 \leq \sum_{n=-\infty}^{+\infty} |\langle f, \varphi_n \rangle|^2 < B\|f\|^2, \quad (2.9)$$

onde as constantes A e B são denominadas *frame bounds*. Quando $A = B$, diz-se que o *frame* é estreito, o que significa que os limites A e B são muito justos. Tem-se que toda base ortonormal é um *frame* estreito com $A = B = 1$, mas nem todo *frame* estreito é uma base ortonormal [9].

O operador de representação R , na representação por *frames* é dado por:

$$R(f) = (f_j)_{j \in \mathbb{Z}}, \text{ onde } (f_j) = \langle f, \varphi_j \rangle. \quad (2.10)$$

De forma geral, em virtude da propriedade de os *frames* não serem necessariamente bases ortonormais, esse operador não é inversível. Apesar disso, é possível reconstruir a função original a partir de sua representação utilizando uma *pseudo-inversa* dada por:

$$f = \sum_{j \in \mathbb{Z}} \langle f, \varphi_j \rangle \tilde{\varphi}_j = \sum_{j \in \mathbb{Z}} \langle f, \tilde{\varphi}_j \rangle \varphi_j, \quad (2.11)$$

onde $\{\tilde{\varphi}_j\}$ é o *frame* recíproco ou *dual frame* do *frame* $\{(\varphi_j)_{j \in \mathbb{Z}}\}$ [9] [10].

Se o *frame* é estreito, então $\tilde{\varphi}_j = A^{-1}\varphi_j$. Nesse caso, a pseudo-inversa é dada por:

$$f = \frac{1}{A} \sum_{j \in \mathbb{Z}} \langle f, \varphi_j \rangle \varphi_j. \quad (2.12)$$

2.1.3 Representação em Multi-resolução

O ser humano percebe o mundo por meio de mecanismos que possibilitam uma visão em múltiplas escalas ou resoluções. Primeiramente, utilizam-se resoluções menores para reco-

nhecer um objeto; depois, resoluções altas, para descobrir detalhes de suas propriedades específicas. Por exemplo, a identificação de certas características de um indivíduo pode ser feita por meio de uma resolução menor, como a cor dos olhos, a cor da pele, o tipo de cabelo etc. No entanto, resoluções altas são necessárias, para que se possam observar detalhes, como a análise das células do indivíduo, por meio de um microscópio, para identificação de doenças.

Assim, é natural que se almeje uma representação de funções em múltiplas resoluções. Por isso é necessário obter uma família de representações, que representem uma dada função em resoluções distintas, e técnicas que possibilitem alternar entre uma resolução e outra. Isso pode ser alcançado por meio do uso da representação aninhada de espaços. Considera-se a seguinte seqüência de sub-espaços fechados $\{V_j\}_{j \in \mathbb{Z}}$:

$$\{0\} \subset \dots \{V_{j-1}\} \subset \{V_j\} \subset \{V_{j+1}\} \dots \subset V, \quad \forall j \in \mathbb{Z}.$$

Esses espaços formam uma estrutura básica de resoluções, de forma que, quanto maior o valor de j , mais próximo de ser igual a V está o sub-conjunto V_j [11].

Seja $P_j : V \rightarrow V_j$, uma família de operadores de representação, tal que $P_j(v) \in V_j$ é a projeção ortogonal de uma função arbitrária de V , v , em V_j , como mostra a Figura 2.1.

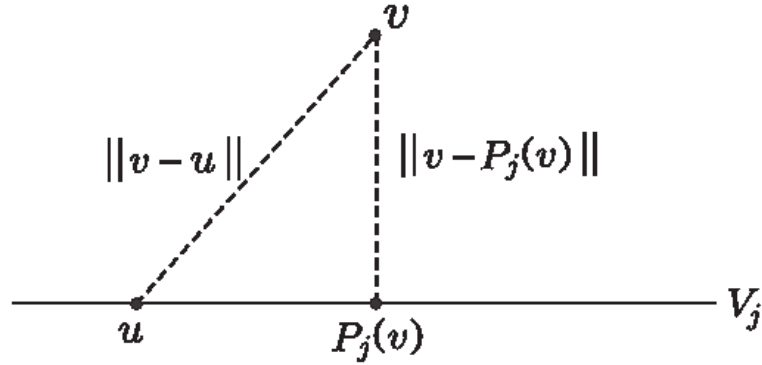


Figura 2.1: O operador de representação por *frames*

Assim, P_j calcula a representação de v no espaço V_j , sendo que esse espaço deve conter funções que aproximam v em um nível de resolução j . Quanto maior o valor de j , maior a resolução das funções de aproximação de f contidas em V_j .

Quanto maior o sub-espaço, maior sua resolução, porque cada espaço $\{V_{j+1}\}$ contém mais detalhes do que o espaço $\{V_j\}$. O operador $Q_j : V \rightarrow V_{j+1}$ estabelece essa afirmação:

$$Q_j(v) = P_{j+1}(v) - P_j(v), \quad (2.13)$$

Se $W_j = Q_j(V_j)$, tem-se que [9]:

$$V_{j+1} = V_j + W_j. \quad (2.14)$$

Ou seja, W_j é o detalhe que deve ser adicionado ao espaço V_j para obter V_{j+1} . Por isso, W_j é denominado *operador de refinamento*. Assim, uma representação mais fina pode ser obtida adicionando detalhes a outra, seguindo esse raciocínio, obtém-se a seguinte equação de decomposição:

$$V_J = V_{j_0} + (W_{j_0} + \dots + W_{J+1}). \quad (2.15)$$

Essa equação pode ser reescrita utilizando operadores de representação:

$$P_J(v) = P_{j_0}(v) + \sum_{j=j_0+1}^J Q_j(v). \quad (2.16)$$

Além disso, v pode ser reconstruído a partir de sua representação de acordo com a seguinte equação:

$$v = \sum_{j \in \mathbb{Z}} Q_j(v) \quad \forall v \in V. \quad (2.17)$$

Ou seja, qualquer função $v \in V$ é a soma de todos os seus detalhes contidos nas diferentes representações. Mas, para que qualquer elemento v possa ser decomposto em representações de diferentes resoluções e reconstruído, há restrições matemáticas que o espaço V e o operador P_j devem atender [9].

A idéia de calcular seqüências que aproximam funções em múltiplas resoluções está intimamente relacionada com *wavelets*.

2.2 A Análise de Funções para a Determinação de suas Características

Um aspecto importante a ser observado é que todas as técnicas citadas na seção 2.1 têm um objetivo comum: representar funções e reconstruí-las a partir de sua representação. De fato, são necessárias técnicas que permitam construir ferramentas de representação (por base, por *frames*, representação em multi-resolução etc.) que sejam flexíveis, concisas e que possibilitem a reconstrução.

Essa necessidade leva ao estudo das variações das funções como uma estratégia para

detectar as suas características específicas, que devem ser levadas em consideração ao representá-las. Ou seja, para criar boas técnicas de representação, é necessário desenvolver ferramentas que possibilitem localizar as características das funções.

Dentre as ferramentas mais tradicionais estão a transformada de *Fourier* e a transformada *wavelet*. Além dessas duas, existem várias outras transformadas, como a de *Laplace*, a de *Walsh*, e a do *Cosseno*, que não são tão populares. A transformada *wavelet*, atualmente, é muito utilizada em diversas aplicações que requerem o tratamento de sinais por causa de suas características de localização no domínio do tempo e da frequência.

2.2.1 O Espectro de Frequência de uma Função e a Exibição de suas Características

Os sinais são caracterizados por funções. O estudo da frequência de uma função é a melhor forma de detectar e analisar as suas características. Por exemplo, ao se escutar uma música, na verdade, ouvem-se variações das frequências do som no decorrer do tempo, pois a frequência caracteriza o agudo e o grave em um sinal de áudio e é nela que é capturada a distinção das cores vermelho e verde em ondas eletromagnéticas.

Seja a função $f = A \sin(2\pi\omega t)$, $A > 0$. O parâmetro A mede a amplitude (valor máximo e mínimo assumido por f), enquanto o parâmetro ω indica quantos ciclos completos do período há no intervalo $[0,1]$. Esse número está diretamente relacionado com o número de oscilações da função em uma unidade de tempo, que é chamado de frequência da função [9]. A Figura 2.2 mostra os gráficos da função f para diferentes valores de ω .

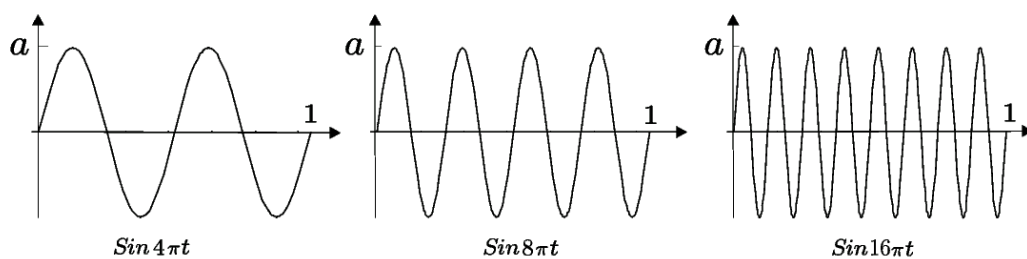


Figura 2.2: Gráficos da função seno com frequência 4, 8 e 16.

Embora a informação da frequência seja de grande importância na análise do sinal, o que normalmente se tem é a sua representação espacial ou temporal. As transformadas possibilitam representar sinais no domínio da frequência. Por meio dessa representação, as transformadas são capazes de revelar as propriedades e as características peculiares das funções. Para tanto, é necessário que sejam capazes de decompor os sinais por meio do

uso de funções bem localizadas no tempo e na frequência. As próximas seções definem as transformadas *wavelet* e de *Fourier*, apresentando suas bases teóricas, os seus aspectos principais e as suas diferenças.

2.3 A Transformada de *Fourier*

A idéia básica de uma transformada é representar funções ou sinais para que a sua informação seja exposta de forma mais simples de se visualizar e manipular. Essa representação ou aproximação da função original é obtida por meio da sobreposição de outras funções. Tal idéia tem origem no trabalho do matemático francês Joseph Fourier, que demonstrou, em 1807, que qualquer função periódica pode ser decomposta como uma soma infinita de senos e cossenos [12].

Seja f uma função periódica com período $T > 0$, ou seja, $f(t+T) = f(t)$, e seja $L_T^2(\mathbb{R})$ o espaço das funções de período T de quadrado integrável, tal que:

$$\int_{t_0}^{t_0+T} |f(t)|^2 dt < \infty. \quad (2.18)$$

A teoria de Fourier estabelece que f pode ser decomposta de acordo com a seguinte equação:

$$f(s) = \sum_{j=-\infty}^{+\infty} a_j e^{i2\pi\omega_j s}, \quad a_j \in \mathbb{R}, \quad (2.19)$$

onde $\omega_j = j/T$ é uma constante. Essa decomposição de f é conhecida como *série de Fourier de f* . Como a família $\{e^{i2\pi\omega_j s}, j \in \mathbb{Z}\}$ é um conjunto ortonormal completo do espaço $L_T^2(\mathbb{R})$, então a Equação 2.19 é uma base ortogonal de representação de f [9].

Essa representação torna mais fácil a análise das frequências presentes em f , pois existe uma frequência fundamental ω da qual todas as outras frequências, $\omega_j, j \in \mathbb{Z}$, são múltiplas. Há também o coeficiente a_j , que mede a amplitude do componente de frequência ω_j na função f . Em particular, se $a_j = 0$, ω_j não está presente na função. Esse coeficiente é calculado utilizando a seguinte equação [9]:

$$a_j = \int_0^L f(u) e^{i2\pi\omega_j u} du. \quad (2.20)$$

Fourier foi motivado a desenvolver suas séries, ao estudar a propagação de calor em corpos sólidos, que é regida por uma equação diferencial linear difícil de ser analisada e resolvida. A característica principal das séries que Fourier obteve é a habilidade de gerar

uma representação exata de uma função, que inicialmente está no domínio do tempo, no domínio da frequência, possibilitando a análise da função nesse domínio, no qual as características das funções são reveladas. Por isso, sua descoberta causou impacto nos campos da Matemática, da Física e da Engenharia, em que essas séries passaram a ser muito utilizadas para soluções analíticas e numéricas de equações diferenciais e também para análise e tratamento de sinais.

Todavia, há um ponto de falha na solução de Fourier: suas séries apenas funcionam para funções periódicas. Mesmo com o sucesso da descoberta de Fourier, levou um século e meio para que os matemáticos que deram continuidade ao seu trabalho compreendessem a convergência de suas séries trigonométricas e completassem a teoria das integrais que constituem a transformada de *Fourier*, estendendo a solução para funções arbitrárias. As próximas subseções descrevem como a teoria das integrais *Fourier* para funções arbitrárias.

2.3.1 A Transformada de *Fourier* no Espaço $L^1(\mathbb{R})$

Primeiramente, a transformada de *Fourier* foi definida no espaço das funções integráveis, $L^1(\mathbb{R})$,

$$L^1(\mathbb{R}) = \{f : \mathbb{R} \rightarrow \mathbb{R}, \int |f(t)| dt < \infty\},$$

e, depois, a definição foi estendida para o espaço $L^2(\mathbb{R})$,

$$L^2(\mathbb{R}) = \{f : \mathbb{R} \rightarrow \mathbb{R}, \int |f(t)|^2 dt < \infty\},$$

das funções de energia finita [11] [10].

No caso do espaço $L^1(\mathbb{R})$, o raciocínio inicia-se a partir da Equação 2.20, que calcula a amplitude de uma frequência. Considerando $s = \omega_j$ e que essa variável pode assumir qualquer valor, obtém-se uma nova equação:

$$a(s) = \int_{-\infty}^{+\infty} f(u) e^{i2\pi su} du. \quad (2.21)$$

Nessa equação, para cada s , $e^{i2\pi su}$ é uma função periódica de frequência s , $s \in \mathbb{R}$, conhecida como função de modulação. A operação $f(u)e^{i2\pi su}$ é conhecida como modulação da função f . Para cada s , a Equação 2.21 pode ser interpretada como uma média ponderada infinita da função f , em que o exponencial $e^{i2\pi su}$ é a função que atribui o peso [9].

Assim, o cálculo da frequência por meio do processo de modulação pode ser enten-

dido da seguinte forma: quando f possui oscilações de frequência de valores iguais ou muito próximos ao valor de s , sua frequência está em ressonância com a frequência da função exponencial. O resultado disso é que $a(s)$ assume valores diferentes de zero. Caso contrário, quando as oscilações de f são completamente distintas das frequências da função exponencial, $a(s)$ assume valores iguais ou próximos de zero.

Portanto, $a(s)$ mede a ocorrência da frequência s na função f , como s varia continuamente no conjunto dos números reais, pode-se dizer que $a(s)$ descreve a densidade de frequência da função f . Quando $a(s) \neq 0$ significa que a frequência s ocorre em f e seu valor mede essa ocorrência.

A equação da transformada de *Fourier* das funções que pertencem ao espaço $L^1(\mathbb{R})$ é obtida mudando a notação da Equação 2.21. Embora a notação das equações seja diferente, a mesma interpretação associada a $a(s)$ na Equação 2.21 cabe à $\hat{f}(s)$ na Equação 2.22. Seja a função $f \in L^1(\mathbb{R})$, sua transformada de *Fourier*, denominada $\hat{f}$, é dada por:

$$\hat{f}(s) = \int_{-\infty}^{+\infty} f(u) e^{-i2\pi su} du. \quad (2.22)$$

Se $f \in L^1(\mathbb{R})$, então essa integral de *Fourier* converge e [10]:

$$|\hat{f}(s)| \leq \int_{-\infty}^{+\infty} |f(u)| du. \quad (2.23)$$

Com isso, tem-se que a transformada de *Fourier* é uma função de s contínua e limitada. Além disso, se $\hat{f}$ também é integrável, ou seja, $\hat{f} \in L^1(\mathbb{R})$, então a transformada inversa de *Fourier* de $\hat{f}$ existe e é dada por [9]:

$$f(t) = \hat{f}^{-1} = \int_{-\infty}^{+\infty} \hat{f}(s) e^{i2\pi st} ds. \quad (2.24)$$

Com essa equação, obtém-se a função f a partir de sua transformada de *Fourier*, a reconstrução de f é uma soma de funções periódicas senoidais $e^{i2\pi st}$ sendo que o coeficiente que mede a amplitude de cada função dessa soma é dado pela transformada de *Fourier* $\hat{f}(s)$.

Utilizando a Equação 2.24, prova-se que, se $\hat{f} \in L^1(\mathbb{R})$, então f é contínua [9]. Com isso, conclui-se que a fórmula da transformada inversa não é válida para funções descontínuas. Para eliminar essa lacuna, a transformada de *Fourier* deve ser estendida para o espaço $L^2(\mathbb{R})$.

2.3.2 A Transformada de *Fourier* no Espaço $L^2(\mathbb{R})$

Além de garantir a transformada inversa para funções descontínuas, outros dois aspectos motivaram a extensão da transformada de *Fourier* para o espaço $L^2(\mathbb{R})$. Um deles é que a transformada de uma função integrável não é necessariamente integrável, ou seja, $\hat{f}$ não é um operador do tipo $\hat{f} : L^1(\mathbb{R}) \rightarrow L^1(\mathbb{R})$, por causa disso nem todas as funções do espaço $L^1(\mathbb{R})$ possuem inversa. O outro aspecto é que há grande interesse no espaço $L^2(\mathbb{R})$, uma vez que ele tem uma estrutura mais rica do que o espaço $L^1(\mathbb{R})$, pois possui produto interno definido e é, na verdade, um espaço de *Hilbert* [9].

Tem-se que, se a função $f \in L^2(\mathbb{R})$ mas $f \notin L^1(\mathbb{R})$, sua transformada de *Fourier* não pode ser calculada, utilizando a Equação 2.22, pois $f(t)e^{i\omega t}$ não é integrável. Por isto, ela é definida, aplicando-se limite às transformadas de *Fourier* de funções que estão em $f \in L^1(\mathbb{R}) \cap f \in L^2(\mathbb{R})$. Por definição, a transformada de *Fourier* de f é dada por $\hat{f}$, no seguinte limite [10]:

$$\lim_{n \rightarrow +\infty} \|\hat{f} - \hat{f}_n\| = 0 \quad (2.25)$$

Na Equação 2.25, $\hat{f}_n$ é a transformada de *Fourier* de f_n , sendo que f_n é uma seqüência de funções $\{f_n\}_{n \in \mathbb{Z}} \in L^1(\mathbb{R}) \cap f \in L^2(\mathbb{R})$ que convergem para f de tal forma que [10]:

$$\lim_{n \rightarrow +\infty} \|f - f_n\| = 0 \quad (2.26)$$

Por definição, f_n é um subconjunto de funções que representa a função f . Como $f_n \in L^1(\mathbb{R})$, então sua transformada de *Fourier*, $\hat{f}_n$, é bem definida, logo, $\hat{f}$ (Equação 2.25) existe.

Assim, tem-se que a transformada de *Fourier* é um operador linear definido no espaço $L^2(\mathbb{R})$. Seja a função $f(t)$, tal que $f : \mathbb{R} \rightarrow \mathbb{R}$, definida no domínio do tempo ⁴. Ao calcular a transformada de *Fourier* de f , obtém-se uma nova função $\hat{f}_\omega$, definida em $L^2(\mathbb{R})$, tal que para cada valor do parâmetro $\omega \in \mathbb{R}$ $\hat{f}_\omega$ representa a densidade da frequência ω em f .

Normalmente, interpreta-se que as funções f , no domínio do tempo, e $\hat{f}(\omega)$, no domínio da frequência, são duas formas diferentes de representar a mesma função. Pode-se ir da representação no tempo para a representação na frequência e vice-versa, utilizando as equações 2.23 e 2.24, respectivamente. A representação de uma função no domínio da frequência revela as suas propriedades e características peculiares. O conhecimento destas

⁴Considera-se, na maioria das vezes, o domínio do tempo, porém as equações funcionam também para outras unidades. Por exemplo, f poderia estar em função do espaço s .

propriedades é extremamente importante na obtenção de uma boa representação.

2.3.3 O Uso da Transformada de *Fourier* na Representação de Funções

O propósito desta seção é melhorar o entendimento das potencialidades da transformada de *Fourier* sob o ponto de vista da representação e reconstrução de funções. Quando se trata de funções periódicas, a Equação 2.20 disponibiliza informações sobre a frequência das funções, por isso é denominada *equação de análise*.

Por outro lado, a Equação 2.19 possibilita obter a função f a partir de seus coeficientes a_j calculados por meio da equação de análise, por isso é denominada *equação de síntese*. Assim, a equação de análise calcula uma seqüência $a_j, j \in \mathbb{Z}$ que representa a função periódica f , enquanto a equação de síntese é a ferramenta de reconstrução.

Quando a função não é periódica, a Equação 2.22 define a transformada de *Fourier* e desempenha o papel de equação de análise das frequências da função f . A Equação 2.24 escreve a função f como a sobreposição de funções periódicas, desempenhando o papel de equação de síntese. As equações de análise e síntese associadas à transformada de *Fourier* não são discretas como no caso das séries de *Fourier* para funções periódicas [9].

Portanto, essa transformada não é uma ferramenta com a qual se pode representar e reconstruir funções arbitrárias, o que a torna inadequada para essa funcionalidade e leva a questionamentos a respeito da eficácia da transformada de *Fourier* [9].

2.3.4 A Análise de Frequência e a Eficácia da Transformada de *Fourier*

De acordo com a definição da transformada de *Fourier*, se $\hat{f}(s_0) \neq 0$, conclui-se que a função f possui frequências iguais ou próximas a s_0 . Nesse caso, o próximo passo da análise de f consiste em localizar a posição destas frequências no domínio do espaço. Tal localização é extremamente importante em várias aplicações e especialmente no problema da representação de funções. Para entender melhor esse problema, pode-se fazer uma analogia com o funcionamento de um radar: a existência de certa frequência detecta a presença de determinado objeto e a localização da frequência possibilita determinar a posição do objeto.

De forma geral, é impossível localizar frequências no domínio do espaço utilizando a transformada de *Fourier*, porque a função de modulação, que é utilizada (a função exponencial) para medir a densidade da frequência nesta transformada, não é bem localizada,

pois não possui suporte compacto⁵, a integral que define a transformada se estende por toda a linha. Portanto, a única informação que se obtém do fato de $\hat{f}(s_0) \neq 0$ é que a frequência s_0 ou frequências próximas a s_0 estão presentes na função f .

O ponto de maior fragilidade da transformada de *Fourier* na análise de sinais é a sua dificuldade em localizar frequências no domínio espacial. De forma geral, se uma função apresenta mudanças repentinas, como descontinuidades, as altas frequências ocasionadas por essas mudanças são detectadas, mas elas influenciam no cálculo da transformada ao longo de todo o domínio, porque a função exponencial não tem suporte compacto. Por isso, a análise de *Fourier* é mais eficaz no estudo de sinais que não sofrem variações repentinas ao longo do tempo (ou espaço). Esse tipo de sinal é chamado *sinal estacionário*.

Para obter uma transformada com propriedades melhores de localização no domínio tempo-frequência, foi feita uma modificação na definição da transformada de *Fourier*. A transformada obtida, denominada *Transformada Janelada de Fourier*⁶, apresenta resultados melhores do que a transformada de *Fourier* na representação de funções.

2.3.5 A Transformada Janelada de Fourier

O propósito da WFT (*Windowed Fourier Transform*) é melhorar a transformada de *Fourier* para obter uma transformada que possua as seguintes propriedades [9]:

- Ao ser aplicada a um sinal $f(t)$, produza um sinal, $\tilde{f}(t, \omega)$, que esteja definido nos domínios do tempo e da frequência simultaneamente.
- Apenas valores que estejam dentro do intervalo de tempo estabelecido, $f(t)$, tal que $t \leq t_1$, sejam analisados no cálculo da “transformada” $\tilde{f}(t, \omega)$.
- O cálculo da transformada de $f(t)$ em um intervalo de tempo específico não influencie no cálculo da transformada de $f(t)$ em outros intervalos de tempo, ou seja, dado um número real $t_0 > 0$, o cálculo de $\tilde{f}(t, \omega)$ deve depender apenas dos valores de t no intervalo $[t - t_0, t]$.

Matematicamente, as duas últimas propriedades citadas mostram que a função de modulação utilizada para detectar as frequências no cálculo da transformada, $\tilde{f}(t, \omega)$,

⁵Suporte compacto significa que fora do intervalo da sua definição até o infinito, a função assume valor zero.

⁶A Transformada Janelada de Fourier, também é designada como transformada de *Gabor* e como STFT *Short Time Fourier Transform*. Sua designação vem do inglês: *Windowed Fourier Transform*

deve ter os seus valores concentrados em uma vizinhança de t , ou seja, deve ser localizada no tempo.

Um método para obter uma função de modulação localizada no tempo consiste em utilizar uma função auxiliar $g(u)$ que faça com que a função de modulação utilizada na transformada de *Fourier*, $e^{-2i\pi\omega u}$ fique “localizada” em uma certa vizinhança no domínio do tempo [9]:

$$g_{\omega,u}(u) = g(u - t)e^{-2i\pi\omega u} \quad (2.27)$$

A atuação dessa função de localização é ilustrada na Figura 2.3. Se a função g é localizada no tempo, obtém-se a localização desejada da função de modulação $g(u - t)e^{-2i\pi\omega u}$. A partir dessa idéia, a transformada janelada de *Fourier* é definida por [9]:

$$\tilde{f}(\omega, t) = g(u - t)f(u)e^{-2i\pi\omega u} du \quad (2.28)$$

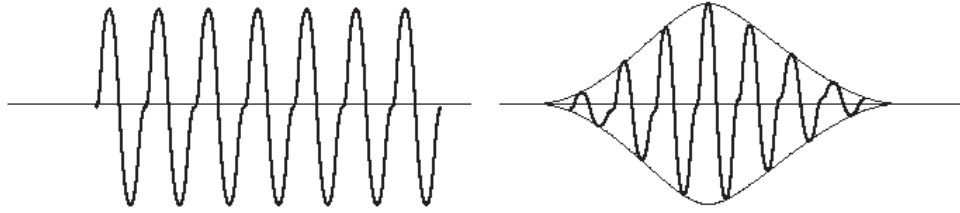


Figura 2.3: Efeito da função de modulação.

Com a WFT, o sinal $f(t)$ é recortado em seções, ou janelas, de forma que a frequência de cada seção pode ser analisada separadamente. Se o sinal tem transições bruscas, efetua-se a janela no sinal de entrada de modo que as seções venham a convergir para zero em seus pontos finais.

Há três aspectos importantes relacionados à WFT que devem ser analisados. Um refere-se à inversibilidade da transformada $\tilde{f}(\omega, t)$. A WFT é inversível e a sua inversa é dada por [9]:

$$f(u) = \frac{1}{\|g\|^2} \iint_{\omega, t} g(u - t)e^{2i\pi\omega u} \tilde{f}(\omega, t) d\omega \quad (2.29)$$

Outro aspecto refere-se ao escopo das funções representáveis por meio da transformada WFT. Dada uma função f , sua transformada janelada de *Fourier*, dada por $\tilde{f}(\omega, t)$, é uma função de duas variáveis. Tem-se que se $f \in L^2(\mathbb{R})$ então $\tilde{f}(\omega, t) \in L^2(\mathbb{R}^2)$. Porém, a imagem da função $\tilde{f}$ não cobre todo o espaço $L^2(\mathbb{R}^2)$ [9].

Uma vez que resolver o problema da representação de funções é o objetivo principal do uso das transformadas, é muito importante analisar a eficiência da WFT na solução desse problema. A WFT possui boas propriedades de localização no domínio tempo-frequência, funciona como um radar, de acordo com a analogia feita anteriormente, sendo capaz de detectar frequências de uma função e localizá-las no tempo.

Assim, a WFT possibilita a análise de funções no domínio tempo-frequência, o que a torna uma boa ferramenta na representação de funções. No entanto, surge um novo aspecto a ser analisado: quão precisamente a WFT pode localizar uma informação da função que ela analisa no domínio tempo-frequência?

Uma análise precisa das informações de uma função $f \in L^2(\mathbb{R})$ pode ser obtida utilizando uma função g com boas propriedades de localização no domínio tempo-frequência (ω, t) . Entretanto, existe um limite para a precisão dessa localização, decorrente de um princípio que governa as transformadas nesse domínio, conhecido como o princípio da incerteza.

De acordo com o princípio da incerteza, não é possível obter uma localização precisa no domínio do tempo e no da frequência simultaneamente. A explicação para isso é simples, para medir frequências é necessário observar o sinal ao longo de um período de tempo, quanto maior for a precisão das medidas de frequência, maior deve ser o intervalo de tempo a ser observado. Assim, se a função g é bem localizada na frequência, o período de tempo T deve ser grande, conseqüentemente g não possui boa localização no tempo. O mesmo raciocínio se aplica quando g é bem localizada no tempo [10].

A WFT utiliza uma escala dada pela largura da janela e analisa o sinal do ponto de vista dessa escala. Se o sinal tem detalhes de frequência fora dessa escala, haverá problemas em sua análise:

- Se os detalhes do sinal são muito menores do que a escala ou a largura da janela, haverá um problema similar ao da transformada de *Fourier*: os detalhes serão detectados, mas a transformada não poderá localizá-los.
- Se os detalhes são maiores do que a escala, eles não serão detectados apropriadamente.

Para resolver esse problema deve-se utilizar uma transformada que seja independente da escala, que utilize uma escala variável ao invés de uma escala fixa. A transformada *wavelet*, apresentada na próxima seção, possui essa propriedade.

2.4 A Transformada *Wavelet*

A transformada *wavelet* resolve o problema da transformada janelada de *Fourier*, utilizando escalas variáveis ao invés de uma escala fixa. A escala é definida pela largura da função de modulação, portanto, para que a escala seja variável, essa função deve ter largura variável. Além disso, a função deve ter boa localização no domínio do tempo. As próximas sub-seções apresentam a definição de funções *wavelets*, como são geradas famílias dessas funções e a definição da transformada *wavelet*.

2.4.1 O Que É *Wavelet*?

As *wavelets* são funções matemáticas que possuem algumas propriedades específicas. De forma geral, estas funções satisfazem as seguintes condições [11]:

1. Deve possuir energia concentrada e finita:

$$E = \int_{-\infty}^{+\infty} |\psi(t)|^2 dt < \infty, \quad (2.30)$$

onde E é a energia da função $\psi(t)$.

2. Deve exibir oscilação no tempo.

Seja $\hat{\psi}(f)$ a transformada de *Fourier* da função ψ , dada por:

$$\hat{\psi}(f) = \int_{-\infty}^{+\infty} \psi(t) e^{-i(2\pi f)t} dt \quad (2.31)$$

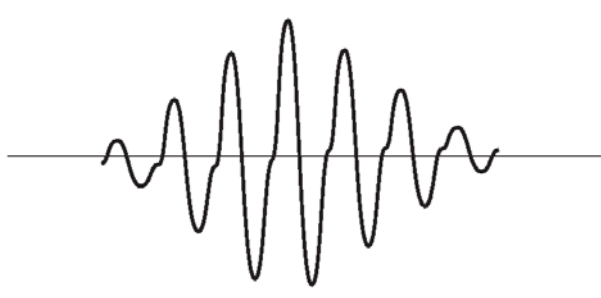
então, a seguinte condição deve ser satisfeita:

$$C_\psi = \int_0^{+\infty} \frac{|\hat{\psi}(f)|^2}{f} df < \infty \quad (2.32)$$

A equação 2.32 é denominada condição de admissibilidade e C_ψ constante de admissibilidade. Para que a função ψ satisfaça essa condição, ela não pode possuir componente de frequência zero, ou seja, se $\psi(u)$ é contínua, então $\hat{\psi}(0) = 0$ [6]:

$$\int_{-\infty}^{+\infty} \psi(u) du = 0. \quad (2.33)$$

Geometricamente, essa condição exige que o gráfico da função ψ oscile de forma que as áreas positivas e negativas se cancelem [9], como mostra a Figura 2.4.

Figura 2.4: Gráfico de uma *wavelet*.

Essas condições, necessárias para que uma função seja considerada uma *wavelet* é que deram origem a esse nome. Literalmente, a palavra *wavelet* significa “onda pequena” e é oriunda do termo francês “*ondelette*”. A condição **1** justifica o termo “pequeno” que caracteriza a função *wavelet*, pois exige que a onda seja bem localizada no domínio do tempo e da frequência. A condição **2**, por sua vez, justifica o termo “onda”, pois a função tem que oscilar para que sua média seja nula. Tem-se, assim, a origem do nome “onda pequena” ou “wavelet” [11]. A Figura 2.5 mostra algumas *wavelets* muito utilizadas na prática.

Figura 2.5: Algumas *wavelets*.

2.4.2 Famílias de Funções *Wavelets*

Para que a transformada *wavelet* seja capaz de resolver o problema da transformada janelada de *Fourier*, é necessário obter uma função de modulação que apresente escala e localização no tempo variáveis. Para obter funções com estas características, inicia-se com uma função $\psi(t)$. Variando a escala dessa função, obtém-se uma família de funções.

Assim, para $p \geq 0$, para todo $s \in \mathbb{R}, s \neq 0$, tem-se [9]:

$$\psi_s(u) = |s|^{-p} \psi\left(\frac{u}{s}\right) = \frac{1}{|s|^p} \psi\left(\frac{u}{s}\right) \quad (2.34)$$

Se ψ tem largura T , então a largura de ψ_s é sT . A modulação da função ψ pelo fator $\frac{1}{|s|^2}$ aumenta sua amplitude enquanto sua escala s diminui e vice-versa. Em termos de frequência, para valores menores da escala s , ψ_s tem altas frequências, e a medida que s aumenta a frequência de ψ_s diminui [9]. Isso está ilustrado na figura 2.6.

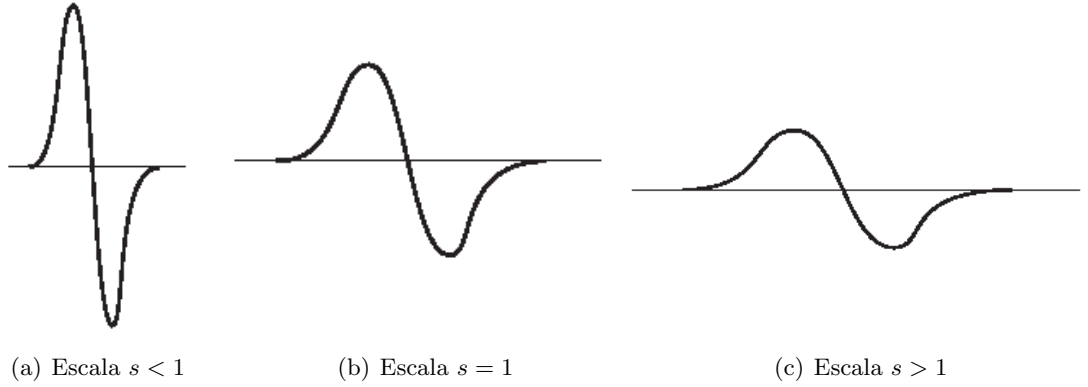


Figura 2.6: Escalas de uma função.

Assim como na transformada janelada de *Fourier*, é necessário localizar cada função ψ_s no tempo. Por isto, para cada $t \in \mathbb{R}$, tem-se que:

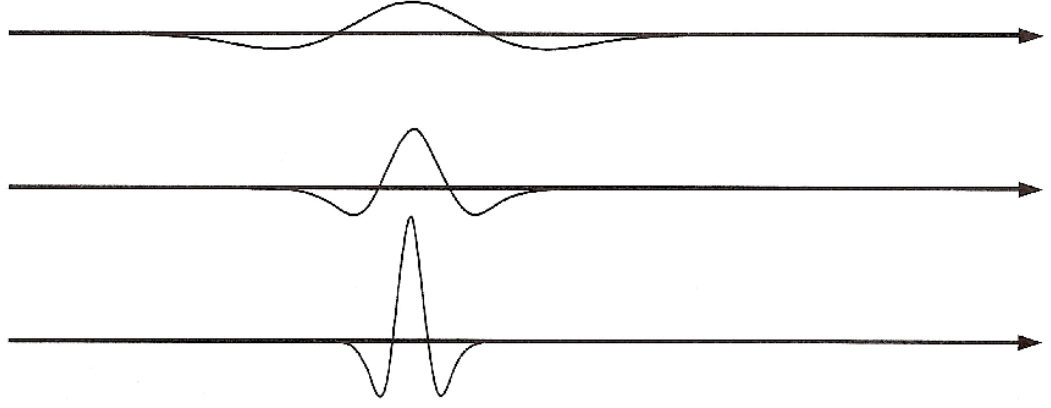
$$\psi_{s,t}(u) = \psi_s(u - t) = |s|^{-p} \psi\left(\frac{u - t}{s}\right) = \frac{1}{|s|^p} \psi\left(\frac{u - t}{s}\right), \quad (2.35)$$

onde $p = \frac{1}{2}$. Considerando que $\psi \in L^2(\mathbb{R})$, então $\psi_{s,t} \in L^2(\mathbb{R})$ e

$$\|\psi_{s,t}\| = |s|^{1-2p} \|\psi\|^2. \quad (2.36)$$

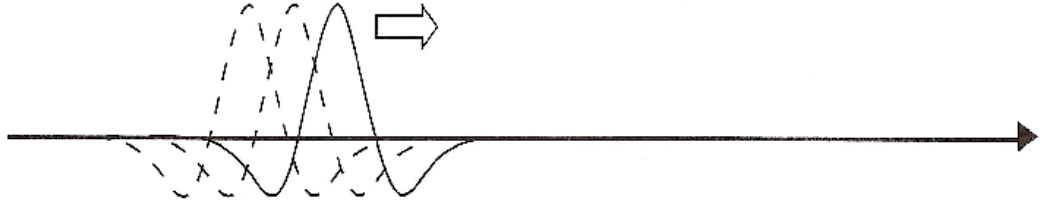
A Equação 2.35 possibilita que uma função *wavelet* ψ , conhecida como *wavelet* mãe ou função de base, gere uma família de *wavelets* por meio de duas operações: a de translação e a de dilatação [6] [10]. A dilatação é determinada pelo valor de s e consiste em expandir e contrair a *wavelet* mãe, permitindo aumentar ou diminuir sua amplitude, como mostra a Figura 2.7.

A translação é determinada pelo valor de t e consiste em movimentar a *wavelet* mãe ao

Figura 2.7: Dilatação da *wavelet* chapéu mexicano.

longo do eixo do tempo. A Figura 2.8 mostra a translação da *wavelet chapéu mexicano*:

$$\psi(t) = (1 - t^2)e^{-\frac{t^2}{2}}. \quad (2.37)$$

Figura 2.8: Translação da *wavelet* chapéu mexicano.

Por meio dessas duas operações, pode-se deslizar a função *wavelet* sobre o sinal a ser analisado e expandi-la ou contraí-la de forma que haja correspondência entre a função e o sinal. Isso é o que a transformada *wavelet* faz. Matematicamente, essa transformada é definida no espaço $L^2(\mathbb{R})$. A equação que a define utiliza uma família de *wavelets*, $\psi_{s,t}$, como funções de modulação, é feita a convolução⁷ entre a função modulação e a função a ser analisada [9]:

$$\tilde{f}(s, t) = \int_{-\infty}^{+\infty} f(u)\psi_{s,t}(u)du. \quad (2.38)$$

A transformada *wavelet* funciona como um microscópio matemático [6], onde t define

⁷Convolução é uma integral que expressa a quantidade de uma função g que se sobrepõe à outra função f à medida em que g é transladada sobre f . Matematicamente, a convolução é definida como um produto entre duas funções, f e g em um dado intervalo, que normalmente é infinito. A equação que define essa operação é a seguinte: $f * g \equiv \int_{-\infty}^{+\infty} f(\tau)g(t - \tau)d\tau$

o local, no eixo do tempo, que está sendo “visto” e s define a ampliação. Por meio do deslocamento associado com a dilatação ou contração da *wavelet* mãe, torna-se possível a mensuração da associação de uma *wavelet* a uma dada função que está sendo analisada.

Para uma determinada escala s , em uma determinada localização t , os segmentos em que a *wavelet* e a função são ambas positivas ou negativas, resultam em uma contribuição positiva no cálculo da integral da Equação 2.38, ao passo que nos segmentos em que os sinais da *wavelet* e da função são opostos, resultam em uma contribuição negativa no cálculo da integral.

Dessa forma, como pode ser observado na Figura 2.9, nos pontos em que há um alto nível de semelhança entre a *wavelet* e a função analisada, os resultados do cálculo da transformada terá valores maiores. Entretanto, nos pontos de baixa similaridade entre a *wavelet* e a função analisada, os valores resultantes do cálculo da transformada são menores.

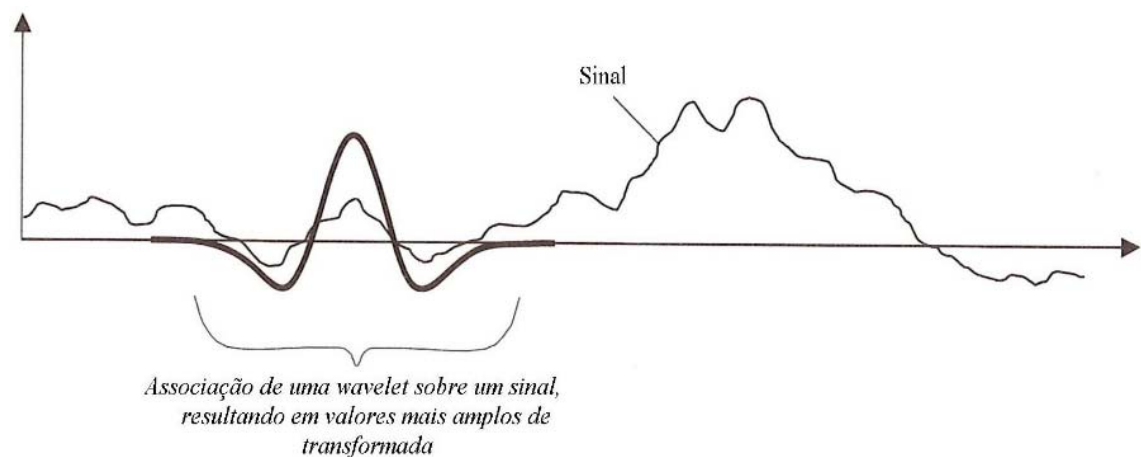


Figura 2.9: Similaridade entre uma *wavelet* e uma função.

Os exemplos apresentados mostram uma *wavelet* a uma escala e uma posição específicas para ilustrar o significado da transformada. Contudo, a transformada *wavelet* contínua não é calculada para escalas arbitrárias e localizações isoladas, mas sim para intervalos contínuos de valores de s e t .

2.4.3 A Transformada *Wavelet* Inversa

Um aspecto importante relacionado à transformada *wavelet* refere-se à sua inversibilidade. Se uma função ψ satisfaz a condição de admissibilidade, então a transformada *wavelet*

$\tilde{f}(s, t)$ é inversível e a Equação 2.39 reconstrói a função f a partir de sua transformada [9]:

$$f(u) = \frac{1}{C} \iint |s|^{2p-3} \tilde{f}_{s,t}(u) \psi_{s,t}(u) ds dt. \quad (2.39)$$

Assim, por meio da transformada *wavelet*, uma função f pode ser decomposta como uma sobreposição de funções $\psi_{s,t}(u)$ e, por meio da transformada inversa, a função f pode ser reconstruída a partir de sua transformada $\tilde{f}(s, t)$.

2.4.4 A Transformada *Wavelet* Discreta

Antes de iniciar a teoria relacionada à discretização da transformada *wavelet* é interessante diferenciar o discreto do contínuo, uma vez que essa diferenciação é causa de muita confusão. Há uma metáfora, utilizada por Scheinerman, que ilustra a diferença [13]:

A diferença é perfeitamente ilustrada pelos relógios de pulso. A matemática contínua corresponde aos relógios analógicos, que separam os ponteiros das horas, minutos e segundos. Os ponteiros se movem suavemente ao longo do tempo. Do ponto de vista de um relógio analógico, entre 12:02h e 12:03h há um número infinito de diferentes tempos possíveis, à medida que o ponteiro dos segundos percorre o mostrador [13].

No entanto, a matemática discreta é comparável a um relógio digital que mostra as horas, os minutos e os segundos. Nesse caso, há apenas um número finito possível de tempos diferentes entre 12:02h e 12:03h, pois esse relógio não reconhece frações de segundo. Assim, não há tempo algum entre 12:02:03h e 12:02:04h. O relógio salta de um instante para o próximo [13].

A DWT (Transformada *Wavelet* Discreta) de uma função contínua $f(t)$ é definida com a discretização dos valores de escala s e de localização t . Sabe-se que a operação de dilatação é multiplicativa, e a sua discretização é simples: fixa-se um valor de escala inicial $s_0 > 1$ e obtém-se as escalas discretas:

$$s_m = s_0^m, \quad m \in \mathbb{Z}. \quad (2.40)$$

Valores positivos de m produzem escalas maiores que 1, enquanto valores negativos produzem valores de escala menores que 1.

Quanto à discretização do tempo, é importante observar que se deve obter uma relação entre tempo e escala de forma que seja possível reconstruir uma função f a partir dos coeficientes gerados do cálculo de sua transformada, $\tilde{f}(s, t)$, utilizando essa relação e com

o mínimo de redundância [9].

O valor da escala influencia na largura da *wavelet*, com o aumento da escala a largura da *wavelet* também aumenta e, com isso, abrange um intervalo de tempo maior. Por outro lado, com a diminuição da escala, a largura da *wavelet* diminui e é necessário aumentar a frequência, para que ela continue a abranger o mesmo intervalo de tempo.

Para obter uma correlação exata entre tempo e escala, deve-se levar em consideração a seguinte propriedade da transformada *wavelet*: quando se altera a escala da função f e, simultaneamente, altera-se a escala no domínio do tempo pelo mesmo fator de escala, a transformada *wavelet* não muda. É importante que essa propriedade da transformada seja preservada na sua discretização, a fim de que ela que seja válida para a DWT.

Para alcançar esse objetivo, quando se passa da escala $s_m = s_0^m$ para a escala $s_{m+1} = s_0^{m+1}$, é necessário incrementar o valor do tempo pelo fator de escala s_0 . Assim, escolhe-se um tempo inicial t_0 e obtém-se o comprimento do intervalo de amostragem: $\Delta t = s_0^m t_0$. Portanto, para cada valor de escala s_0 , o valor do tempo é dado por [9]:

$$tm, n = ns_0^m t_0, \quad n \in \mathbb{Z} \quad (2.41)$$

Considerando os valores discretos dos parâmetros escala e tempo definidos, a seguinte equação define a DWT [6]:

$$\psi_{m,n}(u) = \frac{1}{\sqrt{s_0^m}} \psi \left(\frac{u - nt_0 s_0^m}{s_0^m} \right) \quad (2.42)$$

Nessa equação, os inteiros m e n controlam a dilatação e a translação respectivamente; s_0 é um parâmetro inicial de dilatação, seu valor deve ser maior que 1; e t_0 é um parâmetro inicial de localização, seu valor deve ser maior que zero. A transformada *wavelet* de um sinal contínuo $f(u)$, utilizando a forma discreta de *wavelet* definida na Equação 2.42 é dada por [6]:

$$T_{m,n} = \int_{-\infty}^{+\infty} f(u) \frac{1}{\sqrt{s_0^m}} \psi(s_0^{-m} u - nt_0) du, \quad (2.43)$$

onde $T_{m,n}$ são os valores da transformada discreta *wavelet*, calculados em uma grade de valores de escala e de tempo, no índice m, n . Para a DWT, os valores de $T_{m,n}$ são conhecidos como coeficientes *wavelet* discretos.

2.4.5 A Transformada *Wavelet* Discreta e a Análise em Multi-resolução

A análise em multi-resolução de um sinal possibilita a sua decomposição em aproximações sucessivas de resolução cada vez menor, numa seqüência de estágios de processamento consecutivos. A idéia básica desse tipo de análise utiliza bases ortonormais de *wavelets* como uma ferramenta para descrever matematicamente o “incremento na informação” necessário para se ir de uma aproximação grosseira para uma aproximação de maior resolução.

Essa forma de representação utiliza o conceito de espaço vetorial introduzido na seção 2.1.3. Uma análise em multi-resolução consiste numa seqüência de espaços fechados de aproximações sucessivas V_j . Cada subespaço V_j está contido no próximo subespaço V_{j+1} , que contém mais informações que o anterior. Assim, se uma função está contida em um subespaço, conseqüentemente está em todos os subespaços mais finos.

As seções 2.1.1 e 2.1.2 apresentam os conceitos de bases ortonormais e de representação por frames. De acordo com estes conceitos, as *wavelets* ortonormais possuem limites de frames $A = B = 1$, pois toda base ortonormal é um *frame* estreito e a família de *wavelets*, obtida por meio da Equação 2.42, é uma base ortonormal.

As *wavelets* discretas de grades diádicas são aquelas em que os parâmetros s_0 e t_0 valem 2 e 1, respectivamente. Esse arranjo, conhecido como grade diádica, é considerado como a forma mais simples e eficiente de discretização dos parâmetros s_0 e t_0 para propósitos práticos além de garantir a construção de bases ortonormais. Utilizando tal arranjo, a função *wavelet* pode ser escrita como [6]:

$$\psi_{m,n}(u) = \frac{1}{\sqrt{2^m}} \psi \left(\frac{u - n2^m}{2^m} \right) \quad (2.44)$$

Utilizando a *wavelet* de grade diádica da Equação 2.44, a transformada *wavelet* discreta pode ser definida como [6]:

$$T_{m,n} = \int_{-\infty}^{+\infty} f(u) \psi_{m,n}(u) du \quad (2.45)$$

Se $\psi_{m,n}(u)$ é uma base ortonormal, a função original pode ser reconstruída por meio dos coeficientes *wavelets*, $T_{m,n}$, utilizando a transformada *wavelet* discreta inversa como mostra a seguinte equação [6]:

$$f(u) = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} T_{m,n} \psi_{m,n}(u) du \quad (2.46)$$

As *wavelets* discretas ortonormais de grades diádicas são associadas a funções de escala e a funções de dilatação. A função de escala, também conhecida como *wavelet* pai, influencia na suavidade da representação de uma função e tem a mesma forma da função *wavelet* mãe [6]:

$$\phi_{m,n}(u) = \frac{1}{\sqrt{2^m}} \phi(u2^{-m} - n) \quad (2.47)$$

As *wavelets* pai possuem a seguinte propriedade:

$$\int_{-\infty}^{+\infty} \phi_{0,0}(u) du = 1, \quad (2.48)$$

onde $\phi_{0,0}(u) = \phi(u)$. Além disso, essa função é ortogonal a translações de si mesma, mas não a dilatações de si mesma. A convolução entre a *wavelet* pai e a função a ser aproximada $f(u)$ produz coeficientes de aproximação dados por [6]:

$$S_{m,n}(u) = \int_{-\infty}^{+\infty} f(u) \phi_{m,n}(u) du \quad (2.49)$$

Uma aproximação contínua da função $f(u)$ na escala de índice m pode ser calculada, somando-se uma seqüência de *wavelets* pai nessa escala fatorada pelos coeficientes de aproximação, ou seja [6]:

$$f_m(u) = \sum_{m=-\infty}^{+\infty} S_{m,n} \phi_{m,n}(u), \quad (2.50)$$

onde $f_m(u)$ é uma aproximação da função $f(u)$ cuja suavidade depende do valor da escala de índice m . A função $f(u)$ pode ser representada utilizando séries de expansão que utilizam ambos coeficientes: os de aproximação ($S_{m,n}$) e os de detalhes ($T_{m,n}$):

$$f(u) = \sum_{n=-\infty}^{+\infty} S_{m_0,n} \phi_{m_0,n}(u) + \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} T_{m,n} \psi_{m,n}(u) \quad (2.51)$$

Esta equação mostra que a função contínua original é expressa como uma combinação de uma aproximação de si mesma, calculada em uma escala arbitrária de índice m_0 , adicionada a uma sucessão de detalhes: os da escala m_0 até os da escala $-\infty$.

A Equação 2.49 calcula os coeficientes de aproximação de uma função $f(u)$ na escala m , de forma similar pode-se calcular os coeficientes de detalhes de $f(u)$ na escala m por meio da seguinte equação:

$$d_m(u) = \sum_{n=-\infty}^{+\infty} T_{m,n} \psi_{m,n}(u) \quad (2.52)$$

Utilizando os coeficientes de aproximação e de detalhes, pode-se escrever a Equação 2.51 da seguinte forma:

$$f(u) = f_{m_0}(u) + \sum_{m=-\infty}^{m_0} d_m(u) \quad (2.53)$$

Por meio dessa equação, pode-se mostrar que uma aproximação de maior resolução de uma função, por exemplo na escala $m - 1$, pode ser obtida com a adição dos coeficientes de detalhes aos de aproximação, ambos na escala m , que tem menor resolução que $m - 1$. Essa representação é conhecida como representação em multi-resolução em que os detalhes são o “incremento na informação” necessário para se ir de uma aproximação grosseira para uma aproximação de maior resolução de uma função.

A Figura 2.10 mostra a decomposição e reconstrução em multi-resolução de um sinal c_n por meio da transformada *wavelet* discreta.

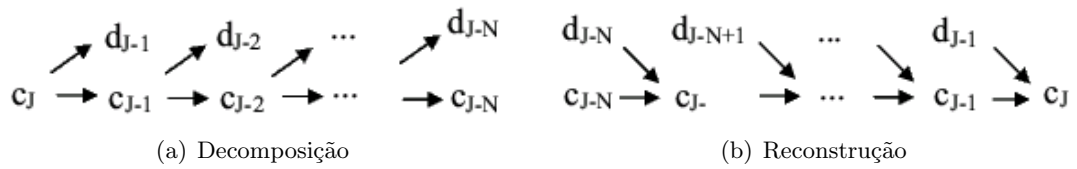


Figura 2.10: Representação em multi-resolução decomposição e reconstrução do sinal c_n .

A Figura 2.10 mostra a decomposição do sinal em coeficientes de aproximação e de detalhes e a sua reconstrução a partir desses coeficientes. Tanto a decomposição quanto a reconstrução são executadas de forma hierárquica, em que se têm escalas de diferentes níveis resolução: quanto menor a escala, maior a resolução.

2.5 Comparação Entre a Transformada *Wavelet* e a de *Fourier*

A principal diferença entre transformada *wavelet* e a de *Fourier* é que as funções *wavelets* são bem localizadas no espaço e na frequência, ao contrário das funções seno e cosseno da transformada de *Fourier*. Essa característica de localidade confere características úteis às tarefas de compressão, detecção de padrão e remoção de ruído em sinais.

Comparada com a WFT, a transformada *wavelet* apresenta vantagem, pela variação das janelas que atuam na localização de discontinuidades do sinal. As *wavelets* mÃe apresentam escala e localização no tempo variáveis, de forma que é possível detectar as discontinuidades. Os valores maiores de escala permitem a análise de baixa frequência do

sinai, ao passo que os menores permitem a análise de alta frequência.

A transformada *wavelet* possui um grande conjunto de funções *wavelets* mãe, ao contrário da transformada de Fourier, que utiliza as funções seno e cosseno. Sendo assim, pela diversidade e numerosidade das famílias de funções *wavelet*, a análise *wavelet* provê acesso imediato à informação, que não pode ser realçada por outros métodos constituídos na relação de Tempo x Frequência, como a análise de *Fourier*.

2.6 O Uso da Transformada *Wavelet* na Inserção de Marca D'água Digital

A transformada *wavelet* permite explorar as estruturas coerentes presentes na maioria dos sinais reais, para construir uma representação ou um mapeamento de grandezas. Nesta representação do sinal, suas estruturas coerentes podem ser distintas em várias escalas, o que possibilita que suas principais características sejam destacadas [7].

Essa forma de representação facilita a análise do sinal e também possibilita sua representação de forma mais sucinta. No caso das imagens, essa característica é extremamente importante, pois possibilita a sua compressão e também a adição de sinais, tal como uma marca d'água, sem danificar o seu conteúdo visual.

A transformada *wavelet* é a mais apropriada para o desenvolvimento do trabalho apresentado nesta dissertação, quando comparada a outras, como a DCT (*Discrete Cosine Transform*) ou a DFT (*Discrete Fourier Transform*), em razão de suas propriedades, apresentadas na seção 2.5, e das características que a fazem amplamente utilizada em processamento digital de imagens, tais como:

- Apresenta boa compatibilidade com o HVS (Sistema Visual Humano - *Human Visual System*);
- A *wavelet* é a tecnologia básica adotada nos padrões JPEG 2000;
- É capaz de oferecer informações de tempo e de frequência simultaneamente;
- Pode ser aplicada a sinais multi-dimensionais;
- Permite a decomposição da imagem em sub-bandas de frequência onde cada sub-banda pode ser quantificada de acordo com a sua importância visual.

2.6. O Uso da Transformada *Wavelet* na Inserção de Marca D'água Digital34

Outra propriedade importante da transformada *wavelet* é a reconstrução perfeita, que é uma condição necessária para que o sinal original, após sua decomposição por meio da transformada, possa ser reconstruído, a partir dos seus coeficientes, após aplicar a transformada inversa.

A escolha da *wavelet* apropriada é extremamente importante para que se obtenha uma melhor aproximação ou representação do sinal original [6]. Nesta dissertação, a *wavelet* escolhida é a CDF (Cohen Daubechies Feauveau) 9/7 bi-ortogonal, pertencente à família CDF [7], cujo gráfico é apresentado na Figura 2.11.

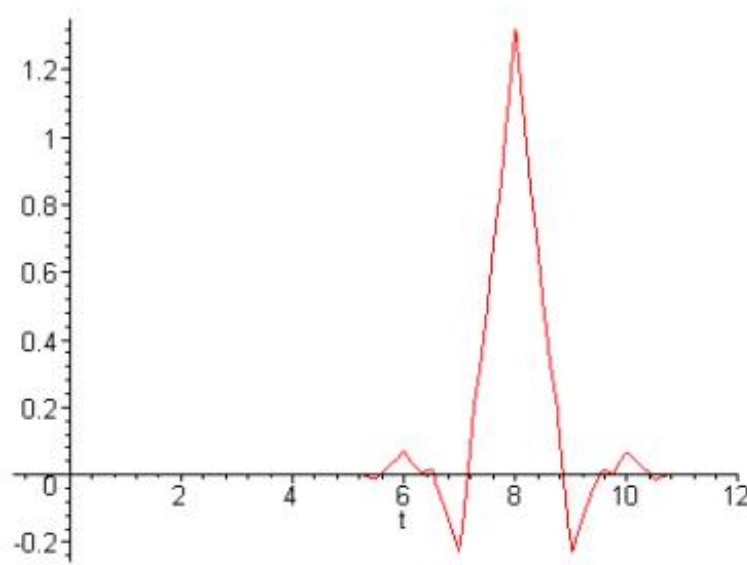


Figura 2.11: Gráfico da função *wavelet* CDF 9/7 bi-ortogonal

Atualmente, a *wavelet* CDF 9/7 é muito utilizada em processamento digital de imagens. Um dos motivos do seu sucesso nas aplicações é que o seu par de filtros é extremamente curto e filtros curtos são bem localizados no tempo. Essa característica está de acordo com os princípios das *wavelets*.

Por causa de suas propriedades de excelente localização no domínio tempo-freqüência e de sua boa compatibilidade com o HVS, essa *wavelet* tornou-se a tecnologia básica nos padrões JPEG2000, em que a CDF 9/7 e a *LeGall* 5/3 são as únicas *wavelets* aceitas. Além disso, o FBI (*Federal Bureau of Investigation*) utiliza a *wavelet* CDF 9/7 na compressão de imagens de impressões digitais [14].

Tanto na compressão de imagens quanto na inserção de marca d'água, é necessário localizar as regiões da imagem em que a modificação do sinal, por eliminação ou adição de informações, não provoque distorções ou alterações visuais perceptíveis à visão humana.

2.6. O Uso da Transformada *Wavelet* na Inserção de Marca D'água Digital³⁵

A *wavelet* CDF 9/7 é eficaz na localização destas regiões, por isso também é amplamente utilizada na elaboração de marca d'água digital. Por causa dessas características, essa *wavelet* é adotada no desenvolvimento do trabalho apresentado nesta dissertação.

A transformada *wavelet* de coeficientes inteiros, denominada transformada *wavelet* inteira, é apropriada para aplicações que requerem processamento de sinal sem perda de informações [8] e é compatível com a maioria das aplicações para processamento de imagens, que normalmente trabalham com valores inteiros. Por isso, a transformada *wavelet* inteira, é utilizada nos processos de decomposição e reconstrução de imagens do esquema de marca d'água proposto nesta dissertação.

O esquema *lifting* é um método para construir *wavelets*. A principal diferença entre esse método e o clássico, via banco de filtros, é que ele não utiliza a transformada de *Fourier* em sua implementação. Além disso, a transformada inversa é facilmente implementada, bastando fazer a operação inversa. Por causa de sua simplicidade, com esse esquema pode-se construir sua própria *wavelet* em casa [15].

Comparado com a implementação via banco de filtros, o esquema de *lifting* apresenta outras vantagens. Uma dessas vantagens é que a complexidade computacional da implementação da DWT é reduzida pela metade, o que torna o processamento mais rápido.

O número de passos de *lifting* necessário na implementação de determinada *wavelet* depende do comprimento do filtro. É provado que para filtros longos a complexidade computacional da implementação da DWT por meio do esquema de *lifting* é a metade da complexidade via banco de filtros, que é $O(N)$ [7] [16]. Outra vantagem é o uso eficiente da memória, pois, em cada passo, a variável que contém o valor calculado no passo anterior pode ser substituída [15].

De acordo com Daubechies, qualquer transformada *wavelet* discreta ou filtros do tipo FIR podem ser implementadas por meio do esquema de *lifting* [7]. Eis porque esse esquema também é muito utilizado para construir as *wavelets* da segunda geração, que são aquelas em que não é necessário transladar e dilatar a *wavelet* mãe.

Além de sua simplicidade e alta velocidade de processamento, outra grande vantagem do *lifting* é que por meio dele, a transformada *wavelet* pode ser facilmente convertida para a transformada *wavelet* inteira, enquanto mantém a propriedade de reconstrução perfeita [16]. No método proposto nesta dissertação, essa técnica é utilizada para implementar a *wavelet* CDF 9/7 bi-ortogonal de coeficientes inteiros.

Como a imagem é um sinal bidimensional, o algoritmo para calcular a transformada

2.6. O Uso da Transformada *Wavelet* na Inserção de Marca D'água Digital36

wavelet discreta bidimensional é utilizado tanto na decomposição quanto na reconstrução da imagem original. Esse algoritmo aplica a DWT uni-dimensional primeiramente nas linhas e, depois, nas colunas da imagem. A Figura 2.12 ilustra a decomposição de uma imagem por meio do algoritmo bidimensional, em cada estágio um FPB (Filtro Passa Baixa) e um FPA (Filtro Passa Alta) são aplicados ao sinal original, cada um desses filtros divide por dois o número de amostras que compõem o sinal.

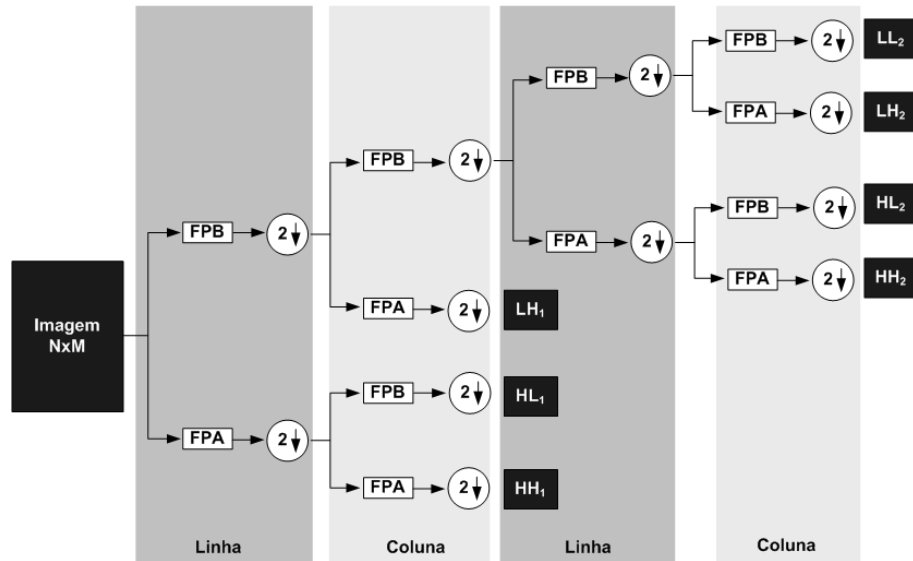


Figura 2.12: Decomposição em dois níveis de um sinal ao aplicar a transformada *wavelet* bidimensional

Ao aplicar a DWT bidimensional na imagem, obtêm-se os coeficientes *wavelet* organizados em sub-bandas de frequência. Os coeficientes *wavelet* resultantes da decomposição da imagem são organizados em uma hierarquia de sub-bandas de frequências. Neste trabalho, a DWT bidimensional nível 3 é aplicada à imagem digital, sendo obtidas as sub-bandas de frequência LL_3 , HL_2 , LH_2 , HH_2 , HL_1 , LH_1 e HH_1 . A Figura 2.13 ilustra as sub-bandas resultantes da decomposição em dois níveis de uma imagem por uma DWT bidimensional.

A sub-banda de frequência de mais baixa, LL_3 , é uma aproximação da imagem original e possui características que garantem que a inserção de uma marca d'água de robustez razoável não provoque distorções perceptíveis na imagem. A marca d'água é inserida na imagem por meio da modificação dos coeficientes dessa sub-banda de frequência. Após a inserção, a transformada inversa é aplicada aos coeficientes e a imagem marcada é obtida.

Esse capítulo descreve o problema da representação de funções e as ferramentas matemáticas que o resolvem, tais como as séries de *Fourier*, a transformada de *Fourier*, a transformada janelada de *Fourier* e a transformada *wavelet*. Tais transformadas são

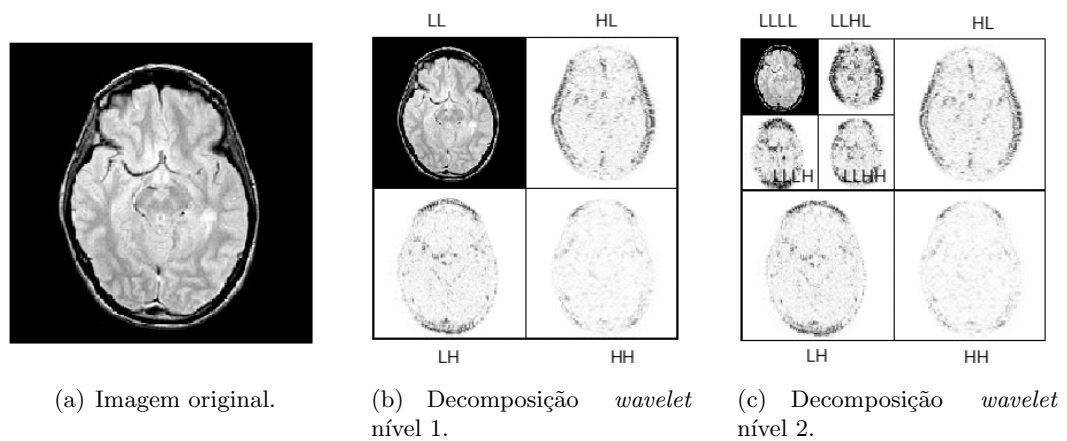


Figura 2.13: Decomposição de uma imagem por meio da DWT

comparadas e as vantagens de seu uso em processamento digital de sinais analisadas.

Também são apresentadas as vantagens do uso da transformada *wavelet* para processamento de sinal e as características da função *wavelet* utilizada, a CDF 9/7, e as vantagens do uso do esquema de *lifting* para implementar a DWT.

O próximo capítulo aborda o problema da autenticação de imagens digitais. São apresentados os possíveis ataques a imagens digitais, o conceito de marca d'água digital, as ferramentas criptográficas capazes de garantir autenticação e assinatura digital e como estas ferramentas podem ser aplicadas juntamente com a marca d'água para oferecer segurança à imagens.

Capítulo 3

Integridade e autenticidade de imagens digitais

Com o crescimento intenso dos sistemas multimídia em rede nos últimos anos, particularmente com o advento da *World Wide Web*, a rede *Internet* tem sido amplamente utilizada para distribuição, comercialização, armazenamento e transmissão de arquivos digitais como imagens, áudio e vídeo.

No entanto, alguns fatores facilitam os ataques aos arquivos de multimídia digitais disponibilizados na rede *Internet*. Um deles é que os canais de comunicação dessa rede são abertos e inseguros, de forma que as informações que neles trafegam ficam sujeitas a diversos ataques, como interrupção, interceptação, modificação ou inserção de informações falsificadas, por parte de entidades não autorizadas.

Outro fator é que, atualmente, existem diversas ferramentas específicas para manipulação de arquivos de multimídia digitais, com as quais, intrusos maliciosos conseguem, facilmente, fazer cópias ilegais ou alterar o conteúdo desses arquivos.

Esses fatores impõem desafios enormes à manutenção da segurança dos arquivos de multimídia digitais, em particular de imagens, com relação a determinados aspectos tais como reconhecimento de propriedade para estabelecimento de direitos autorais, proteção contra cópia ilegal e garantia de integridade e de autenticidade.

Normalmente, há um valor, que pode ser de ordem monetária ou não, agregado aos arquivos de multimídia disponibilizados. Algumas aplicações médicas, por exemplo, disponibilizam imagens provenientes de equipamentos, como as tomografias computadorizadas, ressonâncias magnéticas e ultrassonografias, para serem processadas em ambiente distribuído.

Há, também, uma variedade de sítios, como os de empresas de publicidade, de compra e venda de mercadorias e acervos digitais, que disponibilizam os seus arquivos de multimídia com diversas finalidades, tais como publicação de notícias, exibição de mercadorias ou obras artísticas, entre outros. Portanto, é fundamental dispor de meios para impedir ou detectar ataques como cópia ilegal, falsificação e adulteração, garantindo segurança.

Este capítulo aborda o problema da insegurança na rede *Internet*, os arquivos que trafegam por seus canais ficam sujeitos a diversos ataques. Apresenta os possíveis ataques a imagens digitais e a solução para oferecer segurança contra esses: a marca d'água digital. E, finalmente, descreve como as ferramentas criptográficas podem ser aplicadas juntamente com a marca d'água para garantir a integridade e a autenticidade de imagens digitais.

3.1 Técnicas de Autenticação e Assinatura Digital

Antes da existência dos computadores, a segurança das informações era garantida por meios físicos e administrativos, como, por exemplo, o uso de salas e arquivos fortes para manter documentos sigilosos trancados. Com o advento dos computadores, surgiu a necessidade de criar mecanismos e ferramentas automáticos para proteger as informações e os arquivos neles armazenados.

Outra mudança que afetou o aspecto da segurança foi o surgimento das redes de computadores, em particular da *Internet*, que trouxe as facilidades da comunicação e a necessidade de criar medidas de segurança para proteger as informações durante a sua transmissão, evitando ataques contra a sua autenticidade, integridade e confidencialidade.

Nas redes de computadores, especificamente na *Internet*, diversos protocolos importantes controlam o envio e o recebimento de informações. Dentre eles, os mais utilizados são os da pilha de protocolos conhecida como TCP/IP (*Transport Control Protocol/Internet Protocol*), em que os principais protocolos são o TCP, que fornece comunicação lógica entre processos de aplicações que rodam em hospedeiros diferentes, e o IP, que fornece um serviço de repasse e endereçamento [17].

Uma característica do meio de comunicação da rede *Internet* e dos protocolos TCP/IP é que eles não dispõem de mecanismos de segurança, por isto, no contexto em que há partes comunicantes através da *Internet* de alguns dos ataques possíveis são [18]:

1. **Divulgação:** Revelação de conteúdos de mensagens¹ a qualquer pessoa ou processo

¹Neste capítulo, a palavra mensagem é utilizada de forma genérica para designar qualquer tipo de

que não possui a chave criptográfica apropriada para revelar o seu conteúdo. Há várias técnicas para executar esse tipo de ataque, dentre elas o uso do ataque força bruta para descobrir a chave criptográfica e a criptoanálise do sistema criptográfico para revelar informações secretas.

2. **Análise de tráfego:** Descoberta de padrões no tráfego entre partes comunicantes. Quando a aplicação utiliza o protocolo orientado a conexão, TCP, é possível determinar a duração e frequência das conexões. Além disso, é possível determinar o número e o tamanho das mensagens, quando se utiliza tanto o protocolo orientado a conexão quanto o não orientado a conexão, UDP
3. **Mascaramento:** Inserir mensagens de origem fraudulenta na rede. Por exemplo, um oponente pode criar mensagens, inseri-las na rede e alegar que elas são originadas de uma entidade autorizada; uma pessoa ou um processo, que não seja o receptor, pode gerar reconhecimentos de mensagens recebidas ou não (ACK ou NACK).
4. **Modificação do conteúdo:** Alterar o conteúdo da mensagem, o que inclui inserção, remoção, transposição e modificação.
5. **Modificação da sequência:** Alterar a sequência das mensagens trocadas entre partes comunicantes, o que inclui inserção, remoção e reordenação das mensagens.
6. **Modificação da hora:** Retardar ou duplicar a transmissão de mensagens. Se a aplicação utiliza um protocolo orientado à conexão, uma sequência inteira de mensagens ou mensagens individuais podem ser a retransmissão de alguma sequência válida ou de parte dela previamente transmitida. Se a aplicação utiliza um protocolo não orientado à conexão, uma mensagem individual (por exemplo, um datagrama) pode ser atrasada ou retransmitida.
7. **Repúdio:** Negação do recebimento ou transmissão de uma mensagem por um destino ou origem, respectivamente.

Nessa lista, os ataques 1 e 2 estão relacionados com a confidencialidade de mensagens. As ferramentas criptográficas que lidam com esses ataques devem garantir que a informação transmitida só possa ser descoberta por partes autorizadas e devem ocultar a existência de objetos que possibilitem a análise de tráfego. Essas ferramentas são as que

cifram e decifram mensagens, tais como o RSA e o DES [18], cuja especificação foge ao escopo desta dissertação.

Para garantir a originalidade e a integridade dos dados recebidos por meio da *Internet*, normalmente são utilizadas técnicas de autenticação e de assinatura digital. É importante diferenciar essas técnicas e delimitar os ataques que são evitados por meio de uma delas.

As técnicas que garantem a autenticação de mensagens lidam com os ataques de 3 a 6 da lista de possíveis ataques, já as que garantem a assinatura digital, além desses, lidam também com o ataque 7. Em outras palavras, as técnicas de autenticação de mensagens verificam se uma mensagem vem realmente da origem alegada e se não foi alterada após sua transmissão. Já a assinatura digital é uma técnica de autenticação de mensagens que inclui medidas para garantir que as mensagens não possam ser repudiadas, tanto na sua origem quanto no seu destino.

3.2 Códigos Autenticadores

As técnicas de autenticação e de assinatura digital devem ser analisadas em dois níveis: um nível inferior, constituído de funções que produzem um código autenticador, que é um valor a ser utilizado para autenticar a informação; e um nível superior, constituído de protocolos que possibilitam a um receptor verificar a autenticidade de informações, utilizando o código produzido pelas funções do nível inferior. Esta seção apresenta e analisa as funções do nível inferior das técnicas de autenticação e assinatura digital.

As funções que produzem um código autenticador estão divididas em três classes:

1. **Mensagem cifrada:** a mensagem cifrada, ou embaralhada segundo um algoritmo, é utilizada como código autenticador.
2. **MAC (*Message Authentication Code*):** é utilizada uma função pública da mensagem e uma chave secreta para gerar um código de tamanho fixo que serve como autenticador.
3. **Função *hash*:** Uma função pública que mapeia uma mensagem de qualquer tamanho em um código *hash* de tamanho fixo que serve como autenticador.

As próximas seções descrevem como as funções de cada uma dessas três classes são utilizadas como código autenticador. Nestas seções, alguns símbolos são utilizados: C denota a função criptográfica de chave secreta que cifra mensagens; E a função criptográfica

de chave pública que cifra mensagens; A a origem e B o destino da mensagem; K a chave secreta que apenas A e B conhecem; S um valor secreto que apenas A e B conhecem; K_{PA} a chave privada de um usuário A ; e K_{UA} a chave pública de um usuário A .

3.3 O Uso de Mensagens Cifradas como Código Autenticador

Ao cifrar uma mensagem usando uma chave secreta K , que apenas a origem A e o destino B conhecem, além de confidencialidade, garante-se também autenticidade, pois, se alguém que não conhece a chave tentar alterar a mensagem cifrada, o receptor B perceberá, uma vez que a mensagem decifrada seria uma sequência sem significado.

No entanto, há casos em que uma mensagem pode ser qualquer sequência de *bits*, não sendo possível determinar a sua legitimidade por meio de padrões reconhecíveis. Uma solução para esse problema é o uso de estruturas adicionais que possam ser reconhecidas facilmente, mas que não possam ser replicadas sem o conhecimento da chave criptográfica. Por exemplo, pode-se adicionar códigos de detecção de erros, também conhecidos como FCS (*Frame Check Sequence*), ou *checksum* a cada mensagem antes de cifrá-la.

Essa estrutura redundante também deve estar presente quando se utiliza a criptografia de chaves públicas com propósitos de autenticação. Nesse caso, se A cifra a mensagem M acrescida de um código redundante com sua chave privada e a envia a B , além da confidencialidade e autenticação, tem-se a garantia de assinatura digital, pois A não pode negar o envio da mensagem. A Tabela 3.1 mostra casos em que se utiliza criptografia de chave secreta e pública com vários propósitos, incluindo de autenticação. Nela, M denota a mensagem acrescida do código redundante.

Tabela 3.1: O uso básico de mensagens cifradas

<i>Mensagem enviada</i>	<i>Tipo da chave</i>	<i>Segurança obtida</i>
$C_K(M)$	Secreta	Confidencialidade e autenticação
$E_{K_{UB}}(M)$	Pública	Confidencialidade
$E_{K_{PA}}(M)$	Pública	Autenticação e assinatura digital
$E_{K_{UB}}(E_{K_{PA}}(M))$	Pública	Confidencialidade, autenticação e assinatura digital

3.4 O Uso de MAC como Código Autenticador

Uma técnica de autenticação alternativa utiliza uma chave secreta para gerar um código conhecido como *checksum* criptográfico ou código MAC, que é anexado à mensagem a ser enviada [18]. Nessa técnica as duas partes comunicantes devem compartilhar uma chave secreta, K . Para enviar uma mensagem, A calcula o MAC como uma função de M e de K : $MAC = F_K(M)$, anexa esse código à mensagem, obtendo $M \parallel MAC$, e envia a B . O receptor, por sua vez faz o mesmo cálculo, utilizando a mensagem M recebida e chave K para gerar um novo MAC. O MAC recebido é comparado ao novo calculado. Por meio dessa técnica, se apenas A e B conhecem a chave secreta K , pode-se garantir que:

1. O receptor pode ter certeza de que a mensagem não foi alterada. Se um intruso altera a mensagem mas não altera o MAC, então o MAC que o receptor calcula será diferente daquele anexado à mensagem. Como o interceptor da mensagem, o intruso, não conhece a chave secreta, ele não pode gerar um novo MAC que corresponda à mensagem alterada.
2. O receptor pode ter certeza de que a mensagem foi enviada por A , pois ninguém mais conhece a chave secreta, o que impossibilita a geração de um MAC apropriado por outrem.
3. Se a mensagem inclui um número de seqüência, o receptor pode ter certeza de que a seqüência das mensagens não foi alterada, pois um intruso não conseguiria alterar o número de seqüência sem que o receptor percebesse.

Uma função MAC é semelhante a uma função que cifra mensagens. A diferença é que o algoritmo do MAC não precisa ser reversível, pois não há necessidade de decifrar a mensagem. A Tabela 3.2 mostra formas básicas de utilizar o MAC com propósitos de autenticação. Nesta tabela, K , K_1 e K_2 são chaves secretas compartilhadas por A e B , e F é a função criptográfica que gera o MAC.

Tabela 3.2: O uso de MAC para autenticação de mensagens

Mensagem enviada	Tipo da chave	Segurança obtida
$M \parallel F_K(M)$	Secreta	Autenticação
$C_{K_2}(M \parallel F_{K_1}(M))$	Secreta	Confidencialidade e autenticação
$C_{K_2}(M) \parallel F_{K_1}(C_{K_2}(M))$	Secreta	Confidencialidade, autenticação e assinatura digital

Um dos objetivos do trabalho proposto nesta dissertação é possibilitar a verificação pública, por isto o MAC não é apropriado, pois os algoritmos que geram MACs utilizam chaves secretas.

3.5 O Uso de Código *Hash* Como Autenticador

As funções *hash*, tais como MD5, SHA-1, RIPEMD, dentre outras, produzem um código conhecido como *one-way message digest* ou *código hash*. Para que uma função *hash* H possa ser aplicada com a finalidade de autenticação, deve possuir as seguintes propriedades [18]:

1. Não pode haver restrições do tamanho da mensagem M passada como argumento da função *hash*, ou seja, deve ser possível aplicar H a um bloco de dados de qualquer tamanho.
2. H produz como saída um código de tamanho fixo.
3. Dada uma mensagem qualquer M , $H(M)$ deve ser relativamente fácil de computar, de forma que a implementação por *hardware* e *software* seja prática.
4. Para qualquer código *hash* h , é computacionalmente impossível encontrar M tal que $H(M) = h$. Esta propriedade é denominada caminho único ou *one-way*.
5. Para qualquer mensagem M , é computacionalmente impossível encontrar $N \neq M$ tal que $H(N) = H(M)$. Esta propriedade é denominada resistência à colisão fraca ou *weak collision resistance*.
6. É computacionalmente impossível encontrar um par (M, N) tal que $H(N) = H(M)$. Esta propriedade é denominada resistência à colisão forte ou *strong collision resistance*.

As três primeiras propriedades são requisitos para a aplicação prática de funções *hash* para autenticação. A propriedade 4 garante que seja fácil calcular o código *hash* de uma mensagem, mas impossível calcular a mensagem a partir de seu código. Essa propriedade é importante se a técnica de autenticação envolve o envio de um valor secreto. Se a função H não possui a propriedade caminho único, alguém pode descobrir o valor secreto a partir do código *hash*.

A propriedade 5 garante é impossível encontrar outra mensagem N que gere um código *hash* igual ao de uma dada mensagem M . Na ausência dessa propriedade, alguém poderia executar a seguinte sequência de passos: primeiro, interceptar uma mensagem com seu código *hash* cifrado anexado; segundo, calcular um código *hash* decifrado a partir da mensagem; terceiro, calcular uma mensagem que produzisse o mesmo código *hash*; e quarto, anexar a mensagem calculada ao código *cifrado*.

A propriedade 6 está relacionada com a resistência da função *hash* ao ataque popularmente conhecido como o ataque do aniversário. Isso significa que, dada uma função *hash* H que gera um código de n bits, há 2^n códigos de saída possíveis. De acordo com o ataque do aniversário, se H é aplicada a K mensagens randômicas, o valor de K para que a probabilidade de que pelo menos uma entrada, N , satisfaça $H(N) = H(M)$ seja igual a 0,5, é $2^{\frac{m}{2}}$. O ataque do aniversário se baseia no paradoxo do aniversário, segundo o qual, o valor mínimo de K para que a probabilidade de que pelo menos duas pessoas em grupo de K pessoas seja igual a 0,5 é $2^{\frac{m}{2}}$.

Além dessas propriedades, é necessário que o código *hash* seja dependente da mensagem, ou seja, para uma mensagem M , a saída da função H , $H(M)$, deve estar em função de todos os bits da mensagem, de forma que a alteração de um único bit resulte em uma mudança no código *hash*.

A Tabela 3.3 mostra como um código *hash* pode ser utilizado para autenticação de mensagens.

Tabela 3.3: O uso de função *hash* para autenticação de mensagens

Mensagem enviada	Tipo da chave	Segurança obtida
$C_K(M \parallel H(M))$	Secreta	Confidencialidade e autenticação
$M \parallel E_K(H(M))$	Secreta	Autenticação
$M \parallel E_{K_{PA}}(H(M))$	Pública	Autenticação e assinatura digital
$C_K(M) \parallel C_K(E_{K_{PA}}(H(M)))$	Pública	Confidencialidade, autenticação e assinatura digital
$M \parallel H(M \parallel S)$	-	Autenticação
$C_K(M \parallel H(M \parallel S))$	Secreta	Confidencialidade e autenticação

O trabalho desenvolvido nesta dissertação utiliza o código *hash* como autenticador, a função *hash* escolhida é a RIPEMD-160. As principais características e aspectos de segurança dessa função são descritos na sub-seção a seguir.

3.5.1 A Função *Hash* RIPEMD-160

O algoritmo para cálculo de *hash* RIPEMD-160 foi desenvolvido por um grupo de pesquisadores que conseguiram desenvolver ataques aos algoritmos MD4 e MD5. O grupo desenvolveu primeiro uma versão que gerava um código de 128 *bits*. H. Dobertin (que não fazia parte do grupo) descobriu alguns ataques e esta primeira versão e participou do desenvolvimento de uma nova versão do RIPEMD, o RIPEMD-160 [18].

O algoritmo RIPEMD-160 recebe como argumento uma mensagem de tamanho arbitrário e produz como saída um código *hash* de 160 *bits*. O coração do algoritmo é um módulo que consiste de dez etapas, cada uma com dezesseis passos. As etapas são arranjadas em duas linhas paralelas, cada uma contendo cinco etapas.

Embora tenham uma estrutura similar, em cada etapa é utilizada uma das cinco funções primitivas diferentes, denominadas $f1$, $f2$, $f3$, $f4$ e $f5$. Cada uma dessas funções utiliza operadores lógicos como E , OU , NAO e $OU - exclusivo$. Nas linhas paralelas as funções são aplicadas em ordem inversa; essa estratégia contribui com o aumento da resistência à colisão.

Aspectos de Segurança do RIPEMD-160

Alguns aspectos de projeto do RIPEMD-160 são relevantes na construção de função *hash* segura. O RIPEMD-160 é semelhante ao MD5 e ao SHA-1, uma vez que todos derivam do MD4. Há algumas diferenças e similaridades relativas a aspectos de projeto que devem ser destacadas.

- **Resistência a ataques de força bruta:** Todos os três algoritmos são invulneráveis a ataques de resistência à colisão fraca. Com 128 *bits*, o MD5 é vulnerável ao ataque do aniversário enquanto o SHA-1 e o RIPEMD-160 são resistentes a esse ataque.
- **Resistência à cripto-análise:** O algoritmo RIPEMD-160 foi desenvolvido especificamente para resistir aos ataques cripto-analíticos conhecidos aos quais as funções *hash* MD4 e MD5 não resistiam. Embora pouco seja conhecido a respeito dos princípios do projeto de SHA-1, essa função também é resistente aos ataques cripto-analíticos conhecidos. O uso de duas linhas paralelas de processamento, que dobra o número de passos, torna o RIPEMD-160 mais complexo, conseqüentemente, sua cripto-análise é mais difícil se comparada à do SHA-1.

- **Velocidade:** Todos os três algoritmos, MD5, SHA-1 e RIPEMD-160, são baseados em soma módulo 2^{32} e operações lógicas *bitwise*. A complexidade adicional dos algoritmos SHA-1 e RIPEMD-160 torna-os mais lentos que o MD5. A Tabela 3.4 mostra um conjunto de resultados obtidos em um Pentium 266 MHz, os algoritmos foram implementados em linguagem C++ [18].

Tabela 3.4: *Performance* relativa das funções *hash* MD5, SHA-1 e RIPEMD-160

<i>Algoritmo</i>	<i>Velocidade (Mbps)</i>
MD5	32.4
SHA-1	14.4
RIPEMD-160	13.6

Por causa dessas características, a função *hash* RIPEMD-160 é utilizada neste trabalho para gerar o código *hash* da imagem digital a ser marcada. Um algoritmo é utilizado para gerar uma assinatura digital a partir do código *hash* gerado.

3.6 Criptografia Assimétrica e as Técnicas de Assinatura Digital

A criptografia assimétrica, que também é conhecida como criptografia de chave pública, surgiu da tentativa de resolver dois problemas relacionados com a criptografia convencional. Um dos problemas é o da distribuição da chave. Embora os pesquisadores buscassem desenvolver algoritmos criptográficos inquebráveis, freqüentemente os usuários eram obrigados a compartilhar sua chave secreta, ficando vulneráveis [18].

O outro problema é o das assinaturas digitais. O uso da criptografia não deveria restringir-se aos propósitos militares, ao contrário disto, a tendência era que se estendesse também aos propósitos privados e comerciais. As mensagens e documentos eletrônicos, que se tornavam cada vez mais comuns, precisariam de uma assinatura que correspondesse à assinatura manual, usada em documentos de papel. Por isso, seria necessário um método que garantisse que uma mensagem digital fora enviada por uma entidade particular.

Um método para resolver esses dois problemas deveria ser radicalmente diferente das abordagens criptográficas anteriores, que se baseavam na criptografia simétrica. Daí surgem os algoritmos assimétricos, que utilizam duas chaves separadas, porém relacionadas entre si: a pública e a privada.

Esses algoritmos têm algumas características importantes. Uma delas é que deve ser computacionalmente impossível determinar a chave privada conhecendo apenas a chave pública e o algoritmo criptográfico. Além dessa característica, alguns algoritmos, como o RSA, permitem que qualquer uma das chaves (a privada ou a pública) seja utilizada para cifrar, desde que a outra seja utilizada para decifrar. Tal característica permite que o algoritmo seja utilizado não só com propósitos de confidencialidade, mas também de autenticação e de assinatura digital.

Assim como a assinatura manual, a impressão digital e a íris são utilizadas para identificar uma pessoa em procedimentos ou transações legais, a assinatura digital é utilizada para identificar entidades eletrônicas em transações via *Internet*. A assinatura digital identifica o originador de dados digitalmente assinados e também garante a integridade desses dados, detectando adulterações.

Os métodos de assinatura digital devem assegurar que um transmissor associe sua assinatura digital inforjável a uma mensagem, para que o receptor possa checar a sua origem e integridade, e devem garantir também o não-repúdio. Esses métodos utilizam os algoritmos assimétricos.

Um dos métodos para criar uma assinatura digital é cifrar a mensagem com a chave privada do originador e utilizá-la como assinatura digital. No entanto, essa abordagem apresenta algumas desvantagens que a tornam impraticável. Uma delas é que, nesse caso, assinatura digital tem tamanho igual ao da mensagem original. Isto dobra o volume de dados a serem transmitidos [18].

Outra desvantagem é que o custo computacional para cifrar e decifrar mensagens utilizando algoritmos de chaves públicas é muito alto, de forma que a *performance* do computador e da rede fica comprometida. Além disto, o volume de texto cifrado com a chave privada do transmissor é muito grande, o que facilita a cripto-análise.

Por causa desses aspectos negativos do uso da mensagem cifrada como código autenticador, muitos dos algoritmos assimétricos, como o RSA, são utilizados juntamente com funções *hash* na elaboração métodos para criar a assinatura digital.

Por exemplo, quando um transmissor A quer enviar uma mensagem assinada para o receptor B , ele calcula o código *hash* da mensagem, $H(M)$, e utiliza um algoritmo assimétrico para cifrá-lo com sua chave privada, K_{P_A} , gerando a assinatura digital, $E_{K_{P_A}}(H(M))$. Esta assinatura é anexada à mensagem M , gerando $M \| E_{K_{P_A}}(H(M))$, e enviada à B . Ao receber a mensagem assinada, o receptor calcula um novo código *hash*, $H(M)'$, e decifra

a assinatura digital utilizando a chave pública de A , $H(M)'' = D_{K_{U_A}}(E_{K_{P_A}}(H(M)))$. Os dois códigos, $H(M)'$ e $H(M)''$ são comparados, se forem idênticos e se apenas A conhece a sua chave, então:

1. O receptor pode ter certeza de que a mensagem não foi alterada. Se alguém altera a mensagem mas não altera o código, então o código *hash* que o receptor calcula será diferente daquele anexado à mensagem. Como o intruso não conhece a chave secreta, ele não pode gerar uma nova assinatura que corresponda à mensagem alterada.
2. O receptor pode ter certeza de que a mensagem foi enviada por A , pois ninguém mais conhece a chave secreta, o que impossibilita a geração de uma assinatura digital apropriada por outrem. Assim, A não pode negar o envio da mensagem.
3. Se a mensagem inclui um número de seqüência, o receptor pode ter certeza de que a seqüência das mensagens não foi alterada, pois um intruso não conseguiria alterar o número de seqüência sem que o receptor percebesse.

3.6.1 A Assinatura Digital RSA

Os dois algoritmos de assinatura digital mais utilizados são o DSA (*Digital Signature Algorithm*) e o RSA. No RSA, a assinatura digital é o código *hash* da mensagem a ser assinada cifrado com a chave privada, essa assinatura digital é anexada à mensagem a ser enviada. No lado receptor o código *hash* da mensagem recebida é calculado e comparado ao código decifrado com a chave pública do transmissor

Diferentemente do RSA, o DSA não cifra o código *hash* com chave privada e nem o decifra com a chave pública. Ao invés disto, o DSA utiliza funções matemáticas especiais para gerar a assinatura digital a partir de um código *hash*, da chave privada, de um conjunto de parâmetros globais, compartilhado por um grupo de comunicantes, e de um número randômico, k . A assinatura resultante é composta de dois números de 160 *bits*, r e s . No lado receptor, o processo de verificação é mais complexo que o do RSA. O código *hash* da mensagem é gerado e passado como parâmetro de uma função de verificação, juntamente com a chave pública, e com o conjunto de parâmetros globais compartilhado. Caso a assinatura seja válida, a saída da função de verificação deve ser igual a r .

A segurança do RSA é baseada na dificuldade de fatorar um número muito grande, n , que é o produto de dois fatores primos muito grandes, $n = p * q$. Já a segurança do DSA

é baseada na dificuldade de se calcularem logaritmos discretos².

Os dois algoritmos, RSA e DSA, são considerados igualmente fortes. No entanto, o DSA requer o uso do SHA-1 para garantir uma assinatura digital forte. O processo de verificação de assinatura do DSA é mais complexo e aumenta significativamente a carga computacional em relação ao processo de verificação do RSA. Devido a esses aspectos, optou-se por utilizar o RSA para gerar e verificar a assinatura digital nesta dissertação.

3.6.2 O Uso de Assinatura Digital para Autenticação de Imagens

As imagens digitais, ao serem transmitidas via *Internet*, ficam sujeitas a adulterações assim como outros tipos de mensagens. Na maioria das vezes, não há interesse em manter a confidencialidade de conteúdos digitais, em particular de imagens. Por outro lado, há grande demanda por técnicas que ofereçam que garantia de autenticidade, integridade, reconhecimento de propriedade para estabelecimento de direitos autorais e proteção contra cópia ilegal desses conteúdos.

Os algoritmos de assinatura digital já foram utilizados em aplicações que oferecem segurança às imagens digitais. Em 1993, uma câmera digital confiável foi proposta por Friedman, um microprocessador inserido na câmera gera assinatura digital das fotografias. A chave secreta é armazenada em segurança no microprocessador enquanto a chave pública é impressa como um número de série na própria câmera. A assinatura digital gerada é utilizadas para provar a integridade das imagens fotográficas [19].

As técnicas de autenticação criptográficas são bem apropriadas para serem aplicadas em métodos de autenticação de imagens, pois permitem avaliar a credibilidade de imagens a serem publicadas em jornais e revistas, ou para serem apresentadas como evidência diante de um tribunal de justiça. Estas técnicas também são frágeis à modificação do conteúdo por causa das propriedades das funções *hash* e de outros algoritmos de criptografia (por exemplo, os que cifram e decifram mensagens). Por essas razões, é interessante adaptar os recursos criptográficos de forma apropriada para que as suas características sejam bem aproveitadas em um método de autenticação de imagens.

A marca d'água digital apresenta características específicas que a tornaram amplamente utilizada para autenticação de imagens, no entanto um problema de grande repercussão é a sua confiabilidade. Por isto, faz-se necessário adotar ferramentas criptográficas,

²Considere a equação $y = g^x \text{ mod}(p)$. Dados g , x e p , y é facilmente calculado. No entanto, g , y e p , em geral é muito difícil calcular x . A dificuldade

como funções *hash* e criptografia de chave pública, na elaboração de métodos de autenticação de imagens por meio de marca d'água digital para que esses sejam confiáveis.

3.7 Marca D'água Digital

A marca d'água digital surgiu de uma antiga técnica criptográfica conhecida como esteganografia. Esta técnica visa a ocultar a existência de uma mensagem, escondendo-a em um texto claro de forma que fique ininteligível sem a necessidade de cifrar a mensagem. A marca d'água digital é uma informação embutida em dados digitais, o hospedeiro, de forma que fique invisível aos olhos humanos. Em sua inserção são utilizadas técnicas de esteganografia.

Atualmente, a marca d'água digital é uma solução para coibir a falsificação e distribuição ilegal de arquivos de multimídia digitais. Esta solução também é amplamente adotada com propósitos de autenticação, proteção de direitos autorais e proteção contra cópia não autorizada. Particularmente, nesta dissertação a marca d'água digital é adotada com o propósito de autenticação de imagens digitais a serem transmitidas por meio da *Internet*.

A marca d'água digital é um sinal portador de informação acrescentado ao conteúdo digital que, posteriormente, pode ser extraído e analisado para que se faça alguma alegação sobre esse conteúdo [3]. A imagem que contém a marca d'água é denominada imagem marcada ou hospedeira. Os esquemas para a obtenção de marcas d'água classificam-se entre robustos, frágeis e semi-frágeis, segundo a dificuldade em remover a marca de uma imagem marcada. Essa classificação geralmente determina a finalidade para a qual a marca d'água é utilizada [4].

A marca d'água robusta é projetada para resistir à maioria dos procedimentos de processamento de imagens. A informação embutida em uma imagem deve ser possível de ser extraída mesmo que a imagem hospedeira sofra alterações bruscas como rotação, mudança de escala, mudança de brilho ou contraste, compressão de diferentes níveis com perda de informações, corte das bordas etc.

Esse tipo de marca d'água deve ser possível de ser extraído, a não ser que a modificação na imagem deteriore sua qualidade a ponto de destruir o seu conteúdo visual, ou seja, a marca d'água robusta inserida em uma imagem deve permanecer detectável, enquanto a imagem modificada continuar visualmente reconhecível e identificável como a original. Por essas propriedades, essa marca d'água normalmente é utilizada para a verificação de

propriedade ou de *copyright* das imagens.

Ao contrário da marca d'água robusta, a marca d'água frágil é extremamente sensível à qualquer modificação na imagem marcada, sendo corrompida por qualquer processamento na imagem. Em outras palavras, essa marca d'água garante que a imagem marcada não seja despercebidamente editada ou adulterada, possibilitando detectar qualquer alteração. Por isso, é utilizada para checar a integridade e a autenticidade de imagens.

A marca d'água semi-frágil é um meio-termo entre a robusta e a frágil, pois sobrevive a algumas alterações na imagem mas é sensível a mudanças radicais como a edição. Este tipo de marca d'água procura distinguir as alterações que modificam uma imagem substancialmente das alterações que não modificam o seu conteúdo visual.

Normalmente, no processo de inserção, extraem-se algumas características da imagem que permanecem invariantes por meio das operações “permitidas” para que a alteração dessas características provoque danificações mínimas na marca d'água. Ao contrário, operações não permitidas devem destruir a marca d'água para possam ser detectadas [3]. Assim, a marca d'água semi-frágil também pode ser utilizada para autenticação possibilitando, por exemplo, que a imagem seja compactada antes de ser transmitida via *Internet*.

Outro critério de classificação de marcas d'água refere-se ao processo de detecção, que pode ser privado ou público. Essa distinção é importante pelas suas implicações de segurança: os algoritmos de detecção privados requerem o compartilhamento de chaves secretas. A mesma chave que é utilizada na inserção também é utilizada na detecção da marca d'água, de forma que a capacidade de verificar a marca d'água privada de uma entidade está associada à capacidade de produzir outras marcas d'água em seu nome.

Assim, as marcas d'água cujo algoritmo de detecção é privado ficam sujeitas aos problemas do compartilhamento de chaves. Por isto, aquelas cujo algoritmo de detecção é público têm maior aplicabilidade. Em contrapartida, os algoritmos de detecção privados são mais simples e eficientes do que os públicos, sendo bastante convenientes em determinadas situações.

As marcas d'água são classificadas ainda como *blind* ou *oblivious*, quando o processo de extração não precisa da imagem original. Além dessas classificações, há vários aspectos dos algoritmos de inserção, construção e inserção que diferem as marcas d'água, que, para autenticação podem ser uma imagem significativa, um logotipo, um sinal digital com propriedades bem definidas ou um código binário construído com base no conteúdo da imagem digital, dentre outros.

Com relação aos algoritmos de inserção e remoção, há várias maneiras de inserir a marca d'água em imagens digitais. Entretanto, é mais razoável inseri-la de forma que não produza distorções perceptíveis aos olhos humanos, ou alterações visuais bruscas, mantendo a compatibilidade com o HVS [5]. O receptor extrai o sinal inserido na imagem marcada, gera um sinal usando o mesmo método que o transmissor e compara esses dois sinais. Nos algoritmos de inserção de marca d'água, o sinal autenticador é “atado” ao conteúdo hospedeiro de forma que seja possível localizar as regiões adulteradas.

A segurança dos métodos de autenticação de imagens por meio de marca d'água é baseada em informações, tais como chaves, localização e tamanho da marca d'água, e algoritmos de inserção e remoção secretos, que o transmissor e o receptor compartilham apropriadamente. Para que uma marca d'água para autenticação cumpra seu propósito, deve detectar qualquer alteração na imagem, no caso das frágeis, deve ser única para cada imagem, assim será impossível inseri-la em outra imagem qualquer, e deve ser impossível de ser removida. Além disso, deve resistir a alguns ataques, que são apresentados na próxima seção.

3.7.1 Ataques Contra Marca D'Água para Autenticação

Um ataque a marca d'água é qualquer processamento que visa a prejudicar a sua detecção ou extração. Geralmente, esses ataques visam a ludibriar o receptor da imagem durante o processo de extração, fazendo-o acreditar que uma determinada imagem não possui uma marca d'água, que a marca d'água inserida possui um outro proprietário, ou que uma imagem adulterada é autêntica.

Os ataques, geralmente, são constituídos por duas fases: a primeira é a estimativa da marca d'água e a segunda é a exploração do resultado da primeira fase para enganar o receptor da imagem marcada [20], criando uma marca d'água apropriada para uma imagem adulterada ou alterando a imagem sem provocar danos na marca d'água, dentre outros ataques possíveis.

A Figura 3.1 ilustra um cenário comum na *Internet*, em que Bob, um usuário, verifica a autenticidade de imagens disponibilizadas por Alice, um servidor. Como o canal da rede *Internet* é inseguro, é possível que Malice, um intruso, observe o tráfego dos dados transferidos entre Alice e Bob, podendo também interferir nesse tráfego. Nesse contexto, um método de autenticação de imagens por meio de marca d'água deve oferecer proteção contra as seguintes modalidades de ataques de Malice:

1. Modificação o conteúdo: adulterar a imagem sem prejudicar a marca d'água para que Bob reconheça uma imagem adulterada como se fosse autêntica;
2. Mascaramento: utilizar a marca d'água de Alice para inserir em outras imagens quaisquer e transmiti-las a Bob para que este as reconheça como se pertencessem a Alice.

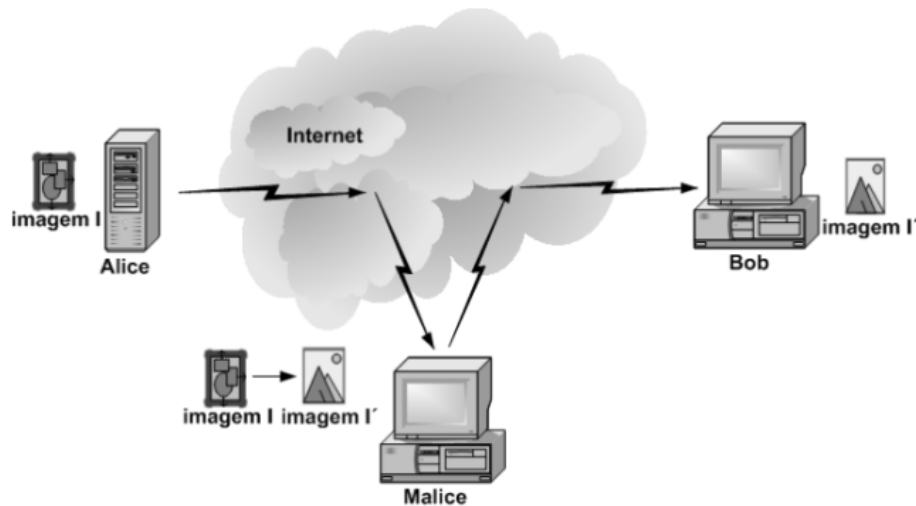


Figura 3.1: Ataque do homem do meio: Malice interfere na comunicação entre Alice (servidor) e Bob (cliente) para forjar a marca d'água

O método apresentado nesta dissertação deve, obrigatoriamente, oferecer proteção contra esses ataques, garantindo autenticação. Adicionalmente, o método dispõe de meios para garantir assinatura digital, evitando que Alice possa negar ser a proprietária de imagens marcadas por ela a Bob e que esse negue o recebimento da imagem de Alice.

Há vários ataques que possibilitam que Malice modifique o conteúdo da imagem ou faça o mascaramento. Para garantir a eficácia de um método de autenticação de imagens digitais por meio de marca d'água, é importante conhecer os potenciais ataques existentes. É extremamente difícil desenvolver uma marca d'água segura contra todos os tipos de ataques, mas é um requisito conhecê-los para que se possa considerá-los nas fases do desenvolvimento e avaliação dos métodos de autenticação de imagens

Um dos ataques mais simples é a modificação casual da imagem marcada, isto é, a imagem é alterada arbitrariamente como se não houvesse marca d'água. Esta forma de ataque deve ser prontamente detectada por qualquer marca d'água para autenticação, pois esse é o tipo mais simples de ataque que um sistema de autenticação de imagens deve tratar.

Uma variação desse ataque consiste em substituir parte de uma imagem por parte de outra, por exemplo, substituir a face de uma pessoa pela de outra. Esse tipo de ataque justifica significativamente a necessidade de uma marca d'água ser capaz de indicar as regiões danificadas em uma imagem adulterada.

Outro tipo de ataque tenta modificar a imagem marcada sem alterar a marca d'água ou criar uma nova marca d'água que o extrator aceite como autêntica. Algumas marcas d'água detectam facilmente uma alteração randômica na imagem, mas falham ao detectar uma modificação cuidadosamente projetada.

Por exemplo, no caso em que uma marca d'água frágil é inserida nos LSB (*Least Significant Bits*) de uma imagem, é bastante provável que uma tentativa de adulterar a imagem, sem levar consideração o local em que foi feita a inserção, provoque uma perturbação na marca d'água e seja detectada. No entanto, alguém pode elaborar um ataque que modifique a imagem sem perturbar os *bits* menos significativos, ou substituir esses *bits* de forma que uma imagem modificada seja considerada autêntica.

Existem ataques que envolvem o uso de uma marca d'água válida obtida de uma imagem marcada para inserir em outra imagem arbitrária. Este tipo de ataque também pode ser executado utilizando apenas uma imagem: primeiro a marca d'água é removida, depois a imagem é modificada e, finalmente, a marca d'água é inserida novamente.

Há também aqueles ataques em que a marca d'água é completamente removida de uma imagem, não deixando vestígios de sua existência. Assim, alguém pode inserir sua própria marca d'água nessa imagem e alegar ser seu proprietário, uma vez que os vestígios que provam que a imagem pertence a outrem foram completamente eliminados. A remoção da marca d'água pode ser feita por meio da adição de ruído à imagem, utilizando técnicas específicas para destruir a marca d'água, ou por meio de análises estatísticas ou ainda por meio de técnicas para estimar a imagem original.

Alguém também pode tentar deduzir a chave utilizada na criação ou inserção da marca d'água. Se a chave é descoberta, então pode-se forjar a marca d'água e inseri-la em qualquer imagem. Há outros tipos de ataques que tentam quebrar o sistema de autenticação, relacionados com os sistemas de autenticação desenvolvidos para canais inseguros cujas vulnerabilidades, muitas vezes, aplicam-se aos sistemas de marca d'água.

A próxima seção apresenta quais são as características básicas que uma marca d'água para autenticação de imagens deve possuir.

3.7.2 Características de Marcas D'água para Autenticação

Esta seção apresenta algumas características das marcas d'água para autenticação de imagens digitais. Embora a importância da presença ou ausência de algumas características dependa da finalidade da aplicação, na maioria dos casos, uma marca d'água deve apresentar as seguintes características [21]:

1. A marca d'água deve ser armazenada, ou inserida, na própria imagem.
2. Detecção de adulteração. Uma marca d'água para autenticação deve detectar, com grande probabilidade de acerto, qualquer adulteração em uma imagem marcada. Essa é uma propriedade fundamental das marcas d'água para autenticação e é um requisito para testar corretamente a autenticidade da imagem. Em alguns casos, também é desejável indicar a quantidade de adulteração ou danos causados à imagem e também a sua localização.
3. Transparência. A marca d'água deve ser invisível ao sistema visual humano.
4. O processo de extração da marca d'água deve ser independente da imagem original. Essa característica, que de acordo com a Figura 1.2 classifica a marca d'água como *blind*, é um requisito importante a ser atendido, pois há casos em que a imagem original não existe ou o seu proprietário tem várias razões para não confiá-la a um terceiro.

Uma dessas razões é que alguém pode inserir a sua própria marca d'água na imagem original e alegar ser seu verdadeiro proprietário. Outra razão é que, comparando a imagem original à marcada, fica mais fácil fazer estimativas que revelam informações sigilosas da marca d'água (por exemplo, há métodos cuja segurança se baseia na localização da marca d'água).

5. O sistema de extração da marca d'água deve ser capaz de localizar e caracterizar as alterações feitas em uma imagem marcada.
6. Os algoritmos de inserção e extração devem ser públicos, ou seja, a segurança da marca d'água deve ser garantida por meio do uso de chaves.
7. A marca d'água inserida em uma imagem, gerada com o uso de uma chave, só deve ser possível de ser extraída mediante o uso da chave de extração correspondente

à chave de inserção. Qualquer outra informação fornecida ao sistema extrator da marca d'água deve provocar falha.

8. A chave utilizada na criação ou inserção da marca d'água deve ser segura e seu espaço deve ser grande o bastante para evitar que seja descoberta por ataques à força bruta. Normalmente, em sistemas que utilizam chaves distintas para inserir e extrair a marca d'água, a chave de inserção é secreta e deve ser impossível de ser descoberta a partir de informações reveladas na extração da marca d'água.
9. A marca d'água deve ser resistente a ataques de inserção ilícita por partes não autorizadas. A seção 3.7.1 menciona o ataque em que a marca d'água é extraída da imagem e substituída por outra.
10. A marca d'água deve ser possível de ser inserida em imagens comprimidas. Embora essa característica não seja de grande importância para todas as aplicações, há casos em que sua presença é uma vantagem significativa.

Obter todas essas características é muito difícil, na prática. A maioria dos algoritmos de marca d'água concentra-se apenas em alguns desses requisitos e, portanto, são aplicáveis a uma gama restrita de circunstâncias.

Outra dificuldade reside no fato de que os requisitos acima podem ser contraditórios, no sentido de que algumas operações em imagens são ora exigidas, ora proibidas. Por exemplo, em uma disputa judicial de autoria ou propriedade legal de uma imagem, é desejável que se possa verificar a marca d'água depois de um ataque de substituição. Deve ser possível mostrar que a marca d'água ainda está na imagem e determinar, inequivocamente, o seu autor ou proprietário. Entretanto, uma verificação de marca d'água deve falhar, indicando adulteração no conteúdo da imagem, após a substituição quando o problema em consideração é exclusivamente o de integridade, como em imagens legais ou médicas.

De fato, o problema da proteção dos direitos autorais e o da verificação de integridade e autenticidade são complementares e, freqüentemente, exigem que técnicas dedicadas sejam aplicadas em sua solução; o primeiro aparece no contexto de codificação e representação de informações, enquanto o segundo está mais diretamente relacionado com técnicas criptográficas [22].

A marca d'água proposta nesta dissertação visa à autenticação de imagens digitais. De acordo com a classificação apresentada na Figura 1.2, é classificada como frágil, é *blind* e de verificação pública. A marca d'água possui todas as características citadas na lista

anterior, exceto que não capaz de localizar as adulterações em todas as circunstâncias, como mostra o teste apresentado na Seção 4.6.2.

3.7.3 Trabalhos Relacionados

Nesta seção, são resumidos os aspectos principais de algumas marcas d'água técnicas que utilizam a DWT para inserção de marca d'água. Recentemente, a marca d'água digital é utilizada para proteção de direitos autorais, proteção do conteúdo, autenticação e detecção de adulterações das imagens digitais. As imagens podem ser representadas no domínio espacial e no domínio da frequência. A mudança da representação de uma imagem, do domínio espacial para o domínio da frequência e vice-versa, é feita por meio do uso de transformadas.

Dentre as transformadas mais utilizadas para processamento digital de imagens, estão a DCT, a DFT e a DWT. A DWT apresenta algumas vantagens de grande relevância em relação às outras transformadas, por isso a maior parte dos pesquisadores a utilizam na elaboração das técnicas de inserção de marca d'água. Nesta seção resumem-se os aspectos principais de alguns métodos de autenticação de imagens digitais por meio de marca d'água.

Dentre os vários métodos de inserção de marca d'água baseados na transformada *wavelet*, o mais conhecido é o de Cox, no qual a marca d'água é inserida nos coeficientes *wavelet*, levando em consideração a sua energia: a robustez da marca d'água é proporcional à capacidade perceptiva e à grandeza do coeficiente [23]. Há muitos outros métodos similares ao de Cox que, entretanto, não estão de acordo com o HVS.

Em um dos métodos pesquisados, é feito o uso de funções caóticas para construir uma marca d'água semi-frágil. A imagem original é decomposta por meio da DWT bidimensional nível 2. A sub-banda de frequência LL_2 resultante da decomposição é utilizada como valor inicial de uma função caótica (*chaotic map*) para construir a marca d'água. A marca d'água obtida é inserida nas sub-bandas LH_2 , HL_2 e HH_2 por meio de quantização par-ímpar, que está de acordo com o HVS. O método é robusto a modificações suaves e sensível aos ataques maliciosos, é capaz de localizar as regiões adulteradas, é invisível e utiliza duas chaves secretas para garantir segurança [24].

Outro método propõe construir uma marca d'água segura por meio de uma DWT de coeficientes inteiros parametrizada. Nesse método, a marca d'água é construída a partir da operação OU exclusivo entre uma imagem binária significativa e uma matriz pseudo-

randômica gerada com o uso de uma chave secreta. A marca d'água obtida é inserida na sub-banda de frequência LL_3 . O método é robusto a compressão JPEG de até 40%, é frágil a alterações maliciosas e é capaz de localizar as regiões adulteradas na imagem [4].

A utilização do esquema de *lifting* diminui a sua complexidade computacional. A segurança da marca d'água é garantida pela chave secreta k e pelo parâmetro α utilizado na construção da transformada *wavelet* [4]. Entretanto, há uma falha de segurança nesse método, pois, se a imagem significativa é conhecida, então é possível obter a matriz randômica secreta, por causa das propriedades da função OU exclusivo. Conseqüentemente, a segurança da marca d'água fica dependendo do parâmetro α .

Yang Zhao propõe o uso de marca d'água dual, sendo uma utilizada para autenticação e outra para compressão de imagens. O método utiliza duas transformadas, a DCT para gerar a marca d'água e a DWT para inseri-la. Uma marca d'água é utilizada para autenticação e detecção de adulterações e outra é utilizada para melhorar a *performance* da técnica de compressão implementada [25].

Nesse método, a ortogonalidade entre as transformadas DCT e DWT é explorada no processo de geração e inserção das marcas d'água. É utilizada uma chave secreta, compartilhada entre o transmissor e o receptor, para permutar a marca d'água para autenticação. Além dessa chave, o método também permite que se utilize criptografia simétrica ou assimétrica para cifrar a marca d'água antes de inseri-la.

Lu apresenta uma marca d'água dual que combina outras duas para resistir a vários tipos de ataques. Não importa qual o tipo do ataque aplicado, uma das marcas d'água sempre é detectada, por isso, elas oferecem proteção dos direitos autorais. Além disso, essa técnica foi melhorada, utilizando valores JND (*Just Noticeable Distortion*), para oferecer também autenticação de imagens [26].

O método proposto por Yueng cria uma imagem binária de tamanho igual ao da imagem hospedeira utilizando uma tabela que mapeia todo valor de luminância possível em 0 ou 1; essa tabela é construída com o uso de uma chave e é conhecida como *look-up table*. A imagem binária resultante é a marca d'água, que é inserida no domínio do espaço da imagem hospedeira trocando os LSB de todos os pixels pelo valor LUT (*look-up table*) correspondente. No lado receptor, a LUT pode ser reconstruída utilizando a chave secreta.

A verificação da integridade pode ser feita por meio da inspeção visual ou por meio de um sistema automatizado de verificação, ambos comparam a marca d'água extraída com a original. Essa marca d'água é muito sensível a alterações na imagem, mas é muito

vulnerável a ataques, pois é possível fazer alterações na imagem sem alterar os *bits* menos significativos. Além disso, a chave utilizada na construção da LUT deve ser compartilhada. Esse método foi melhorado utilizando uma cifra por blocos de 64x64 ao invés da LUT, a marca d'água é inserida em blocos de 32x32 [27].

A proposta de Liu utiliza a sub-banda de baixa frequência, resultante da decomposição da imagem por meio da DWT, para inserir a marca d'água. Esta sub-banda contém a maior parte da energia da imagem e a magnitude de seus coeficientes é maior do que a magnitude dos coeficientes das demais sub-bandas. Isto significa que os coeficientes de baixa frequência possuem maior capacidade perceptual, o que possibilita que a inserção da marca não provoque distorções na imagem.

Nesse método, é feito o uso de uma chave secreta para gerar uma sequência binária pseudo-aleatória, denominada PN , para construir a marca d'água: $W'_i = W_i \oplus PN_i$, W_i é o i -ésimo *byte* da marca d'água. Essa marca d'água é inserida nos coeficientes da sub-banda de baixa frequência por meio da substituição dos cinco LSBs de cada coeficiente [28]. Os algoritmos de inserção e extração de marca d'água da proposta apresentada nesta dissertação são baseados nesse método.

Um aspecto interessante a ser observado é a necessidade de algoritmos de verificação pública de marcas d'água, pois, por meio deles, alegações de propriedade de imagens podem ser aceitas ou rejeitadas, sem que o proprietário precise revelar informações privadas. Esse tipo de algoritmo sugere o uso de sistemas criptográficos assimétricos em sua implementação, como Friedman pioneiramente indicou em 1993 [19].

Apesar da necessidade óbvia de algoritmos de verificação pública, poucos pesquisadores levam em consideração essa necessidade. Friedman propôs uma câmera digital confiável com um microprocessador embutido, para possibilitar a autenticação de imagens utilizando assinatura digital. A chave secreta é armazenada em segurança no microprocessador, enquanto a chave pública é impressa como um número de série na própria câmera.

Para cada fotografia, a câmera gera dois arquivos: a imagem capturada e a sua assinatura digital (o código *hash* da imagem cifrado com a chave privada). Os dois arquivos gerados e a chave pública são utilizados para provar que as imagens fotográficas não sofreram adulteração por meio de um *software* para verificação [19]. No entanto, a assinatura digital é impressa e, portanto, não satisfaz o requisito de estar embutida na própria imagem hospedeira. Assim, o usuário fica responsabilizado por manter os dois arquivos, a imagem e sua assinatura digital, uma vez que ambos são necessários na verificação.

Há outro método em que se propõe a utilizar chave pública na elaboração de marca d'água para vídeo comprimido [29], mas a idéia sugerida no princípio é abandonada, já que a idéia subjacente é tornar pública apenas parte da chave pseudo-aleatória em vez de se adotar diretamente criptografia de chave pública. Dessa forma, o uso da expressão “chave pública” pelos autores não é padrão.

Um outro exemplo de algoritmo de marcas d'água é realmente baseado em criptografia assimétrica, todavia a assinatura digital da imagem é armazenada numa base de dados ou acrescentada ao cabeçalho da imagem e, portanto, não satisfaz o requisito de estar embutida na própria imagem hospedeira [30].

Outro método proposto utiliza, efetivamente, chaves assimétricas. Nesse método, a marca d'água é inserida no domínio espacial da imagem, por meio da substituição dos *bits* menos significativos. No processo de inserção, a marca d'água (de tamanho igual ao da imagem) e a imagem original são divididas em blocos. Os bits menos significativos dos pixels de um bloco da imagem são zerados. Este bloco, juntamente com o tamanho da imagem (definido pelos valores de M e N), são passados como parâmetros de uma função *hash*. O valor *hash* calculado e um bloco da imagem binária utilizada como marca d'água são combinados utilizando a função OU-exclusivo [31].

O bloco resultante dessa combinação é cifrado utilizando a chave privada e inserido nos *bits* menos significativos dos pixels de um bloco da imagem. O processo é repetido até que todos os blocos da imagem estejam marcados. No processo de extração da marca d'água, a chave pública é utilizada para verificar a autenticidade da imagem [31].

Porém, há ataques que esse método não detecta. Por exemplo, é possível construir uma imagem utilizando blocos de imagens autênticas marcadas por meio do método de Wong. Esta construção é feita de tal forma que essa imagem forjada seja aceita, erroneamente, como legítima. Além disto, a marca d'água é inserida nos *bits* menos significativos dos pixels da imagem, sem levar em consideração as características das imagens digitais e o HVS [3].

De forma geral, um problema evidente na maioria dos algoritmos de autenticação de imagens por meio de marca d'água é que são utilizadas chaves simétricas, o que impossibilita a verificação pública e traz como herança todos os problemas relacionados com o compartilhamento de chaves.

Outro problema é que a maioria dos esquemas convencionais de marca d'água publicados são inseguros, de modo que, uma vez que o algoritmo de inserção é conhecido,

um oponente consegue expor e alterar a informação escondida. Assim como Kerckhoff's estabeleceu para as técnicas de criptografia, a segurança das técnicas de autenticação de imagens deve ser garantida não pela confidencialidade do algoritmo, mas sim por meio do uso de chaves [4].

Para contornar esses problemas, é preciso adotar algoritmos públicos e utilizar chaves para garantir a segurança. Além disto, o uso de chaves públicas possibilita que a autenticidade e integridade de imagens digitais sejam verificadas publicamente. A próxima seção apresenta alguns problemas relacionados com autenticação de imagens digitais que devem ser levados em consideração ao elaborar um método de autenticação de imagens digitais.

3.7.4 Aspectos de Segurança de Marca D'água Digital

Há diversos cenários em que as técnicas de autenticação de imagens por meio de marca d'água são aplicadas. Um desses cenários é a distribuição na *Internet* com diversos fins. Normalmente, as aplicações que utilizam marca d'água para autenticação de imagens digitais na *Internet* são do tipo cliente-servidor, havendo um *host* servidor que disponibiliza as imagens marcadas para os *hosts* clientes.

Nesse caso, é inviável utilizar um método de autenticação baseado em criptografia simétrica, em que os clientes devem conhecer a chave que o servidor utiliza na inserção da marca d'água para que possam extraí-la, pois essa chave teria que ser compartilhada entre os vários clientes. A consequência disso é que qualquer cliente poderia inserir a marca d'água do servidor em imagens adulteradas e quaisquer imagens e disponibilizá-las a outros clientes, que não perceberiam o ataque. Nesse caso, nem a origem nem o conteúdo da imagem são íntegros.

Uma solução possível para esse problema é utilizar um protocolo para estabelecer uma chave de sessão a ser compartilhada entre o servidor e um único cliente. Nesse caso, pode-se afirmar que um intruso não conseguirá utilizar essa chave para inserir a marca d'água, porque apenas o cliente e o servidor a conhecem.

Todavia, essa solução apresenta alguns aspectos negativos, como maior complexidade computacional e maior latência. Como os algoritmos para geração de chaves requerem altas taxas de processamento, o aumento do número de requisições simultâneas a esse serviço pode sobrecarregar o servidor. Também deve ser considerado o aumento do número de pacotes transitando no canal de comunicação que liga o servidor à *Internet* por causa do protocolo de estabelecimento das chaves. Dependendo da capacidade do canal e da

complexidade do protocolo, pode ser aumentado significativamente o tempo necessário para que um cliente autentique uma imagem. Além disto, o servidor deverá inserir a marca d'água nas imagens para cada cliente. Todos esses fatos tornam o sistema caro e complexo.

Para sanar essas dificuldades, o método de construção, inserção e extração de marca d'água apresentado nesta dissertação tem por objetivo garantir a segurança e oferecer uma implementação com menor complexidade computacional. Além disto, o método deve ser genérico, podendo ser aplicado em vários cenários, principalmente na autenticação de imagens distribuídas na *Internet*. O uso de ferramentas criptográficas, como função *hash* e algoritmos de geração e verificação de assinatura digital, garantem a segurança do método.

Capítulo 4

Uma Proposta de Assinatura Digital para Imagens por Meio de Marca D'água

Este capítulo apresenta uma proposta de assinatura digital para imagens, utilizando marca d'água. O objetivo principal desta proposta é propiciar segurança àqueles que utilizam a *Internet* como meio de disponibilização de imagens digitais. A segurança, neste caso, refere-se à garantia de que a integridade e a autenticidade das imagens digitais possam ser verificadas por meio de uma marca d'água segura, publicamente verificável, invisível ao HVS e capaz de possibilitar a localização das possíveis adulterações.

A Figura 4.1 apresenta um cenário que fornece uma visão geral da proposta apresentada. Nesse cenário, o transmissor constrói a marca d'água, a insere na imagem, e transmite a imagem marcada via *Internet*. No lado receptor a marca d'água é extraída e as adulterações são detectadas e localizadas.

Para garantir a segurança, a marca d'água possibilita a detecção e, quando possível, a localização de diversos ataques, por exemplo, aqueles que visam à alteração do conteúdo e ao mascaramento. Além disso, a proposta tem como características a baixa complexidade computacional na implementação, conforme está especificado na Seção 4.6.1, que também é um requisito importante.

Para atender aos requisitos de segurança e de verificação pública, o algoritmo de construção da marca d'água utiliza a função *hash* e as funções criptográficas de chave pública. O uso desse tipo de chaves permite que a marca d'água seja publicamente verificável e

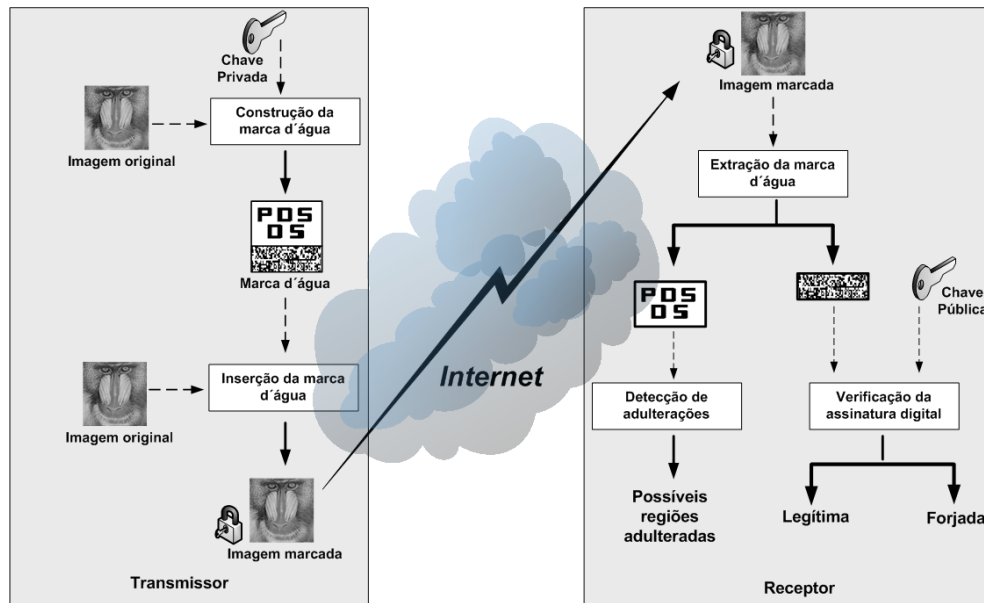


Figura 4.1: Visão geral da proposta apresentada.

contorna os problemas relacionados com o compartilhamento de chaves, conforme está descrito no Capítulo 3.

Para atender ao requisito de invisibilidade da marca d'água, a transformada *wavelet* é utilizada para decompor a imagem digital, pois, dessa forma, possibilita a identificação das regiões em que a inserção da marca d'água não provoca distorções visíveis ao HVS. A função *wavelet* escolhida é a CDF 9/7 bi-ortogonal, por causa das suas propriedades de boa localização no domínio do tempo e da frequência, de compatibilidade com o HVS e dos bons resultados apresentados no processamento digital de imagens. O esquema de *lifting* é utilizado na implementação da transformada *wavelet*, para reduzir a complexidade computacional da implementação.

As próximas seções descrevem o ambiente de desenvolvimento e de testes da proposta implementada, os algoritmos de construção, inserção e extração da marca d'água, o processo de detecção e localização de adulterações na imagem. Também são apresentados os resultados de tempo obtidos com a inserção e extração da marca d'água em diversas imagens, os resultados de qualidade visual das imagens marcadas por meio do PSNR (*Peak Signal-to-Noise Ratio*) e os resultados da simulação de diversos ataques às imagens digitais que a marca d'água deve detectar.

4.1 Ambiente de Desenvolvimento

O computador utilizado na implementação e na realização dos testes possui as seguintes características de *hardware* e *software*:

- Processador AMD Duron com cache de 64KB;
- CPU com frequência igual a 896MHz ;
- Memória RAM (*Randomic Access Memory*) de 367 MB (Mega Bytes);
- Sistema operacional Linux, distribuição *Slackware* versão 11.0.0, *Kernel* Linux-2.6.19 [32] [33] [34];
- Compilador gcc, versão 3.4.6 [35].

No processamento digital de imagens, são utilizadas as funções pertencentes ao pacote de *software* QccPack [36], que contém uma coleção de rotinas para codificação, quantização e compressão de dados. Estão incluídas nesse conjunto de bibliotecas as funções para aplicar a DWT e a IDWT às imagens digitais.

As funções criptográficas pertencentes à biblioteca OpenSSL [37], incluída na instalação padrão do sistema operacional, são utilizadas para o cálculo do código *hash* e para gerar e verificar a assinatura digital. A função *hash* utilizada é a RIPEMD-160(), que gera um código *hash* de 160 *bits*. As funções *RSA_sign()* e *RSA_verify()* são utilizadas para gerar e verificar a assinatura digital, respectivamente.

4.2 Implementação da DWT e da IDWT Utilizando o Esquema de *Lifting*

A transformada *wavelet* possibilita representar funções destacando as suas características para a visualização e manipulação. A idéia básica desta transformada é explorar a estrutura correlacional presente na maioria dos sinais reais para construir a sua representação ou aproximação.

A DWT pode ser construída via banco de filtros ou por meio do esquema de *lifting*, que apresenta algumas vantagens com relação à implementação via banco de filtros, tais como: a complexidade computacional é reduzida pela metade, o que torna o processamento mais rápido; uso eficiente da memória; fácil implementação da transformada inversa e da transformada de coeficientes inteiros; dentre outros.

4.2. Implementação da DWT e da IDWT Utilizando o Esquema de *Lifting* 67

Por causa dessas vantagens, esse esquema é adotado para implementar a DWT nesta dissertação. As subseções a seguir apresentam detalhes da implementação da DWT e da IDWT, utilizando a *wavelet* bi-ortogonal CDF 9/7 de coeficientes inteiros, por meio do esquema de *lifting*.

4.2.1 A Implementação da DWT

A implementação da DWT por meio do esquema de *lifting* consiste em três passos: *Dividir*; *Predizer*; e *Atualizar* [15]. Seja o sinal unidimensional $x = (x_k)_{k \in \mathbb{Z}}, x_k \in \mathbb{R}$. O diagrama de blocos da Figura 4.2 ilustra os três passos que constituem o esquema de *lifting*.

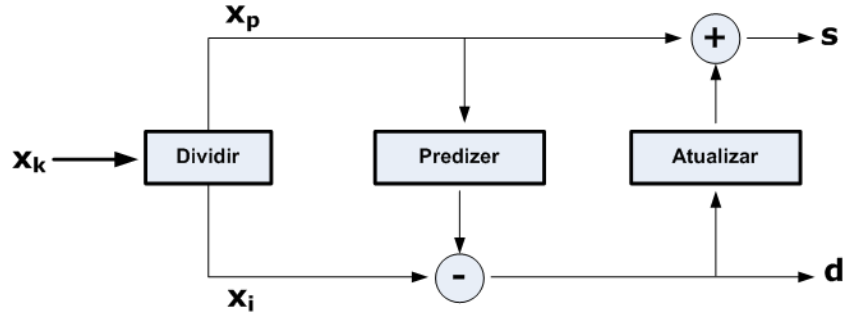


Figura 4.2: Diagrama de blocos dos três passos do esquema *lifting*

No passo *Dividir*, o sinal de entrada, x_k , é dividido em dois conjuntos: um com amostras de índice par, $x_p = (x_{2k})_{k \in \mathbb{Z}}$, e outro com amostras de índice ímpar, $x_i = (x_{2k+1})_{k \in \mathbb{Z}}$. Cada conjunto contém a metade no número de amostras do sinal original [15].

Normalmente, o sinal possui uma estrutura local que correlaciona suas amostras, por isso as amostras dos conjuntos de índice par e ímpar são fortemente correlacionadas. Assim, a partir de um dos conjuntos, é possível prever o outro com precisão razoável. No passo *Predizer*, a estrutura presente no sinal é explorada para prever o conjunto de amostras de índice ímpar a partir do conjunto de índice par, utilizando o operador de predição P . A predição não precisa ser exata, por isso o coeficiente de detalhe, ou de diferença, d , deve ser armazenado, como mostra a Equação 4.1 [15].

$$d = x_i - P(x_p) \quad (4.1)$$

Por exemplo, na *wavelet Haar*, a predição do valor de uma amostra de índice ímpar, x_{2k+1} , utiliza a amostra de índice par vizinha, x_{2k} , como preditor. O coeficiente de detalhe é calculado como a diferença entre a amostra de índice ímpar e seu preditor [15]:

$$d = x_{2k+1} - x_{2k} \quad (4.2)$$

O detalhe d , calculado no passo *Predizer*, possibilita revelar algo da estrutura correlacional presente no sinal. Com esse passo, tem-se uma transformada de (x_p, x_i) para (x_p, d) , porém é necessário obter informações referentes à frequência do sinal [15].

As amostras x_p não possuem a mesma frequência que o sinal original, pois são obtidas no passo *Dividir*, em que o sinal é dividido em dois conjuntos de amostras. Por isso, é necessário substituir as amostras do conjunto de índice par por coeficientes de baixa frequência s , obtidos por meio do operador *Atualizar*, U [15]:

$$s = x_p + U(d) \quad (4.3)$$

Considerando o exemplo da *wavelet Haar*, o operador *Atualizar* é dado por:

$$s_k = x_{2k} + \frac{d_k}{2} \quad (4.4)$$

Os coeficientes s formam uma representação do sinal x_k e os coeficientes de detalhe d são as informações necessárias para obter o sinal original a partir de sua representação. Independentemente de quais sejam os operadores P e U , a predição e a atualização são sempre inversíveis, o que possibilita a reconstrução perfeita do sinal x_k [15].

Os passos do esquema de *lifting* para construir a DWT de coeficientes inteiros, utilizando a função *wavelet CDF 9/7*, a partir do sinal x_k , são [8]:

$$\begin{cases} s_k^{(0)} = x_{2k} \\ d_k^{(0)} = x_{2k+1} \end{cases} \quad (4.5)$$

$$\begin{cases} d_k^{(1)} = d_k^{(0)} + \text{Round}(\alpha(s_k^{(0)} + s_{k+1}^{(0)})) \\ s_k^{(1)} = s_k^{(0)} + \text{Round}(\beta(d_k^{(1)} + d_{k-1}^{(1)})) \end{cases} \quad (4.6)$$

$$\begin{cases} d_k^{(2)} = d_k^{(1)} + \text{Round}(\gamma(s_k^{(1)} + s_{k+1}^{(1)})) \\ s_k^{(2)} = s_k^{(1)} + \text{Round}(\delta(d_k^{(2)} + d_{k-1}^{(2)})) \end{cases} \quad (4.7)$$

$$\begin{cases} d_k^{(3)} = d_k^{(2)} + \text{Round}((\zeta - \zeta^2)s_k^{(2)}) \\ s_k^{(3)} = s_k^{(2)} + \text{Round}((-1/\zeta)d_k^{(3)}) \end{cases} \quad (4.8)$$

4.2. Implementação da DWT e da IDWT Utilizando o Esquema de *Lifting* 69

$$\begin{cases} d_k^{(4)} = d_k^{(3)} + \text{Round}((\zeta - 1)s_k^{(3)}) \\ s_k^{(4)} = s_k^{(3)} + d_k^{(4)} \end{cases} \quad (4.9)$$

$$\begin{cases} s_k = s_k^{(4)} \\ d_k = d_k^{(4)} \end{cases} \quad (4.10)$$

Onde $\text{Round}()$ é uma função do tipo inteiro de arredondamento matemático,

$$\begin{aligned} \alpha &= -1,58615986717275; & \beta &= -0,05297864003258 \\ \gamma &= 0,88293362717904 & \delta &= 0.44350482244527 \\ \zeta &= 1.14960430535816 \end{aligned} \quad (4.11)$$

A Equação 4.5 define a operação de divisão. Nas demais equações, as operações com os coeficientes d_k são de predição, enquanto as operações com os coeficientes s_k são de atualização. Em pseudo-código, esses passos de *lifting* são implementados da seguinte forma:

```
int comprSinal; // Armazena o comprimento do sinal
//Este bloco implementa a predição da Equação 4.6
for (k = 1; k < comprSinal - 2; k += 2)
    x[k] += Round(alpha * (x[k - 1] + x[k + 1]));
x[comprSinal - 1] += Round(2 * alpha * x[comprSinal - 2]);

//Este bloco implementa o Atualizar da Equação 4.6
x[0] += Round(2 * beta * x[1]);
for (k = 2; k < comprSinal; k += 2)
    x[k] += Round(beta * (x[k + 1] + x[k - 1]));

//Este bloco implementa a predição da Equação 4.7
for (k = 1; k < comprSinal - 2; k += 2)
    x[k] += Round(gamma * (x[k - 1] + x[k + 1]));
x[comprSinal - 1] += Round(2 * gamma * x[comprSinal - 2]);

//Este bloco implementa o Atualizar da Equação 4.7
x[0] += 2 * (Round(delta * x[1]));
for (k = 2; k < comprSinal; k += 2)
```

4.2. Implementação da DWT e da IDWT Utilizando o Esquema de *Lifting* 70

```
x[k] += Round( $\delta$  * (x[k + 1] + x[k - 1]));

//Este bloco implementa a predição da Equação 4.8
for (k = 1; k < comprSinal; k += 2)
    x[k] += Round(( $\zeta$  - pow( $\zeta$ ,2)) * x[k - 1]);

//Este bloco implementa o Atualizar da Equação 4.8
for (k = 0; k < comprSinal; k += 2)
    x[k] += Round((-1/ $\zeta$ ) * x[k + 1]);

//Este bloco implementa a predição da Equação 4.9
for (k = 1; k < comprSinal; k += 2)
    x[k] += Round(( $\zeta$  - 1) * x[k - 1]);

//Este bloco implementa o Atualizar da Equação 4.9
for (k = 0; k < comprSinal; k += 2)
    x[k] += x[k + 1];
```

Na implementação da proposta apresentada, é usada a função pertencente ao pacote `QccPack` [36] que implementa esse algoritmo: `QccWAVLiftingAnalysisIntCohen-DaubechiesFeauveau9_7()`. Nessa proposta, é aplicada a DWT bidimensional nível 3 à imagem a ser marcada, denominada *I*.

Na implementação da DWT bidimensional a DWT unidimensional é aplicada primeiramente nas linhas da imagem e, depois, nas colunas. Antes de aplicar a transformada, a imagem *I* é lida para a variável `Image`, que é uma variável do tipo `QccIMGImage`, utilizando a função `QccIMGImageRead()`:

```
QccIMGImageRead(&Image)
```

A estrutura `QccIMGImage` é definida da seguinte forma:

```
typedef struct{
    int image_type;
    QccString filename;
    QccIMGImageComponent Y;
    QccIMGImageComponent U;
    QccIMGImageComponent V;
} QccIMGImage;
```

4.2. Implementação da DWT e da IDWT Utilizando o Esquema de *Lifting* 71

Nessa estrutura, o campo `image_type` armazena o tipo da imagem, `filename` armazena o caminho para o arquivo que contém a imagem, `Y` contém os componentes de luminância, e, `U` e `V`, os componentes de crominância da imagem. O principal campo da estrutura `QccIMGImageComponent` é uma matriz que contém variáveis do tipo *double* que representam os pixels dos componentes da imagem ¹.

A função `QccIMGImageRead()` lê a imagem cujo caminho é especificado no campo `filename`. Primeiramente, chama a função `QccIMGImageDetermineType()` para determinar o tipo da imagem, que pode ser: PPM, PGM, PBM, ICP ou desconhecida, caso a imagem não seja de nenhum destes tipos.

O tipo, ou formato, é determinado por um “número mágico” presente no cabeçalho da imagem ou pela extensão do arquivo. Depois de identificar o tipo, `QccIMGImageRead()` lê a imagem para a variável `Image`; para cada tipo de imagem, há um algoritmo específico para fazer a leitura.

Em seguida, os pixels do componente `Y` da variável `Image` são copiados para a variável `SubbandPyramid`. Esta variável é do tipo `QccWAVSubbandPyramidInt`, definida da seguinte forma:

```
typedef struct
{
    QccString filename;
    QccString magic_num;
    int major_version;
    int minor_version;
    int num_levels;
    int num_rows;
    int num_cols;
    int origin_row;
    int origin_col;
    int subsample_pattern_row;
    int subsample_pattern_col;
    QccMatrixInt matrix;
} QccWAVSubbandPyramidInt;
```

O principal componente da estrutura `QccWAVSubbandPyramidInt` é a matriz que armazena os coeficientes *wavelet*, resultantes da decomposição da imagem por meio da DWT, organizados em sub-bandas. Esta matriz é armazenada no campo `QccMatrixInt matrix`.

¹Os componentes de crominância da imagem é que definem suas cores, quando a imagem é em escala de cinza as matrizes `U` e `V` são nulas

4.2. Implementação da DWT e da IDWT Utilizando o Esquema de *Lifting* 72

Ao decompor uma imagem em K níveis, obtêm-se os coeficientes *wavelet* organizados em $3 \times K$ sub-bandas de detalhes, que são de frequência alta, e 1 sub-banda de frequência baixa, LL_K , que é uma aproximação de baixa resolução da imagem original. A Figura 4.3 mostra a hierarquia de sub-bandas resultante da decomposição *wavelet* nível 3. O número de níveis de decomposição fica armazenado no campo `num_levels` da estrutura `QccWAVSubbandPyramidInt`.

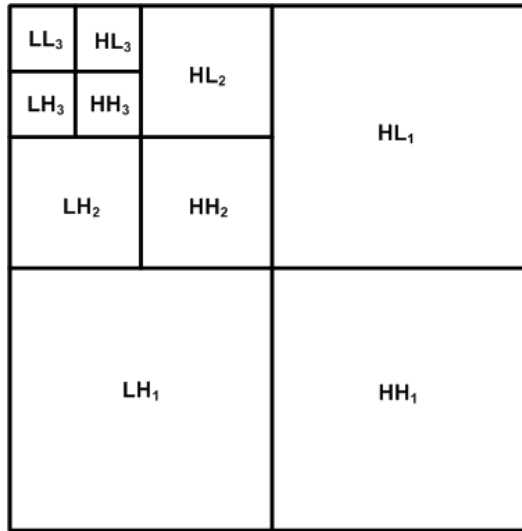


Figura 4.3: Hierarquia de sub-bandas resultante da decomposição *wavelet* nível 3.

Para aplicar a DWT à imagem, também é utilizada uma variável, denominada `Wavelet`, que é do tipo `QccWAVWavelet`. Esta estrutura é definida da seguinte forma:

```
typedef struct {
    int implementation;
    int boundary;
    QccWAVLiftingScheme lifting_scheme;
    QccWAVFilterBank filter_bank;
} QccWAVWavelet;
```

O campo `implementation` indica qual implementação da *wavelet* será utilizada, que pode ser o método clássico via banco de filtros, `QCCWAVWAVELET_IMPLEMENTATION_FILTERBANK`, ou o esquema de *lifting*, `QCCWAVWAVELET_IMPLEMENTATION_LIFTED`. O campo `boundary` indica o tratamento a ser aplicado aos limites do sinal, dependendo de seu comprimento².

²Por exemplo, se o sinal tem comprimento ímpar, pode-se assumir que a sub-banda de baixa frequência tem uma amostra a mais ou a menos que as sub-bandas de alta frequência, dependendo do valor deste campo

4.2. Implementação da DWT e da IDWT Utilizando o Esquema de *Lifting* 73

O campo `lifting_scheme` contém a estrutura `QccWAVLiftingScheme`, caso esse esquema seja utilizado. Caso contrário, o campo `filter_bank` contém a estrutura do banco de filtros, `QccWAVFilterBank`. Dentre os campos da estrutura `QccWAVLiftingScheme`, o de maior relevância é o campo de tipo inteiro `scheme`, que indica qual a *wavelet* a ser aplicada.

Após copiar os pixels da imagem para o campo `matrix` de `SubbandPyramid`, a DWT de coeficientes inteiros bidimensional nível 3, é aplicada a esta variável:

```
QccWAVSubbandPyramidIntDWT(&SubbandPyramid, NumLevels, &Wavelet)
```

Nessa função, o parâmetro `NumLevels` é um valor inteiro que contém o número de níveis de decomposição que, nesse caso, é três. Na variável `Wavelet`, o campo `scheme` da estrutura `lifting_scheme` contém o valor que especifica que a função *wavelet* utilizada é a biortogonal CDF 9-7 de coeficientes inteiros.

A função `QccWAVSubbandPyramidIntDWT()` chama `QccWAVWaveletDWT2DInt()` para executar a DWT na matriz contida no campo `matrix` da variável `SubbandPyramid`. Ao campo `num_levels` desta variável é atribuído o valor de `NumLevels`, que é passado como parâmetro.

A função `QccWAVWaveletDWT2DInt()`, por sua vez, chama a função que faz a análise *wavelet* em um sinal bidimensional, `QccWAVWaveletAnalysis2DInt()`. Essa função é chamada `NumLevels` vezes, passando, como parâmetro de entrada, uma matriz que é a sub-banda do nível de decomposição corrente.

Assim, a decomposição da matriz `matrix` é efetuada recursivamente. Na primeira execução de `QccWAVWaveletAnalysis2DInt()`, a matriz é decomposta em quatro sub-bandas com 1/4 de seu tamanho original: uma de baixa frequência, LL_1 e três de alta frequência, LH_1 , HL_1 , HH_1 . Na próxima decomposição, a sub-banda de baixa frequência, LL_1 , é passada como parâmetro sendo decomposta em quatro sub-bandas com 1/4 do tamanho de LL_1 : LL_2 , LH_2 , HL_2 e HH_2 . Finalmente LL_2 é decomposta nas sub-bandas LL_3 , LH_3 , HL_3 e HH_3 .

Para decompor cada sub-banda, `QccWAVWaveletAnalysis2DInt()` chama a função `QccWAVWaveletAnalysis1DInt()` para cada linha e, em seguida, para cada coluna da matriz recebida como parâmetro. A função `QccWAVWaveletAnalysis1DInt()`, por sua

4.2. Implementação da DWT e da IDWT Utilizando o Esquema de *Lifting* 74

vez, efetua um nível de decomposição em um sinal unidimensional. Essencialmente, esta função chama `QccWAVLiftingAnalysisInt()`, passando como parâmetro um vetor de valores inteiros.

A função `QccWAVLiftingAnalysisInt()` decompõe o sinal unidimensional, recebido como parâmetro, em duas sub-bandas: uma de baixa frequência, retornada na primeira metade do vetor, e outra de alta frequência, retornada na segunda metade do vetor. O algoritmo que implementa os passos de *lifting* que fazem a decomposição do sinal está descrito no início desta seção.

Assim a função `QccWAVLiftingAnalysisInt()` faz a análise de cada linha e, depois, de cada coluna de cada sub-banda em decomposição. Na primeira decomposição, as linhas e, depois, as colunas da matriz `matrix`, que contém os pixels do componente Y da imagem, são analisadas. Na segunda, as linhas e as colunas de LL_1 e, na terceira decomposição, as linhas e as colunas de LL_2 .

O resultado da análise *wavelet* da imagem é uma matriz de coeficientes organizada em uma hierarquia de sub-bandas apresentada na Figura 4.3, que é retornada no campo `matrix` da variável `SubbandPyramid`. Aplicando a IDWT sobre esta matriz, obtém-se a imagem original novamente. A próxima seção descreve a implementação da IDWT.

4.2.2 A Implementação da IDWT

A IDWT é facilmente implementada por meio do esquema de *lifting*, basta inverter as operações dos passos *Dividir*, *Predizer* e *Atualizar* para obter os passos da inversa: *Juntar*, *Desfazer Predizer* e *Desfazer Atualizar*, que são executados em ordem inversa [15]. A Figura 4.4 ilustra os três passos que constituem o esquema de *lifting* que implementa a IDWT.

O passo *Desfazer Atualizar* recupera o conjunto com amostras de índice par, x_p , utilizando os coeficientes de detalhe, d , e dos coeficientes de baixa frequência, s , subtraindo a informação dada no passo *Atualizar*:

$$x_p = s - U(d) \quad (4.12)$$

Considerando o exemplo da *wavelet Haar*, apresentado na seção 4.2, o passo *Desfazer*

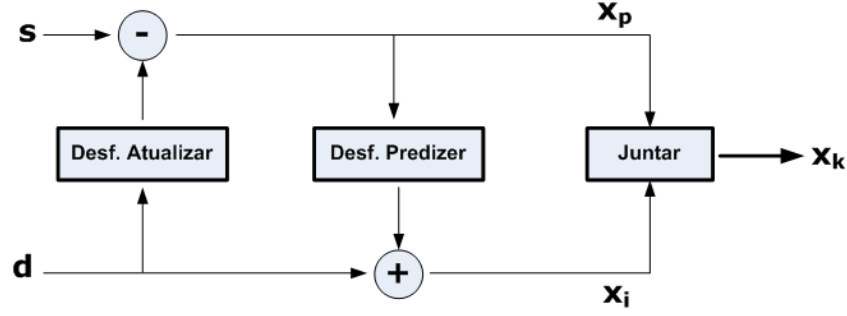


Figura 4.4: Diagrama de blocos dos três passos do esquema *lifting* que implementa a IDWT

Atualizar é dado por:

$$x_{2k} = s_k - \frac{d_k}{2} \quad (4.13)$$

O passo *Desfazer Predizer* recupera o conjunto com amostras de índice ímpar, x_i , adicionando o detalhe à informação obtida no passo *Predizer*:

$$x_i = P(x_p) + d \quad (4.14)$$

No caso da *wavelet Haar*, o passo *Desfazer Predizer* é dado por:

$$x_{2k+1} = d + x_{2k} \quad (4.15)$$

Depois de recuperar o conjunto de amostras de índice par x_p e o de amostras de índice ímpar x_i , o passo *Juntar* recupera o sinal x_k unindo estes conjuntos. Os passos do esquema de *lifting* para construir a IDWT de coeficientes inteiros são facilmente implementados, bastando trocar o operador soma por subtração e a ordem dos passos de *lifting*. Em pseudo-código, os passos que implementam a transformada que utiliza a função *wavelet* CDF 9/7 são os seguintes [8]:

```
int comprSinal; // Armazena o comprimento do sinal
//Este bloco implementa o Desfazer Atualizar
for (k = 0; k < comprSinal; k += 2)
    x[k] -= x[k + 1];
```

4.2. Implementação da DWT e da IDWT Utilizando o Esquema de *Lifting* 76

```
//Este bloco implementa o Desfazer Predizer
for (k = 1; k < comprSinal; k += 2)
    x[k] -= Round(( $\zeta$  - 1) * x[k - 1]);

//Este bloco implementa o Desfazer Atualizar
for (k = 0; k < comprSinal; k += 2)
    x[k] -= Round((-1/ $\zeta$ ) * x[k + 1]);

//Este bloco implementa o Desfazer Predizer
for (k = 1; k < comprSinal; k += 2)
    x[k] -= Round(( $\zeta$  - pow( $\zeta$ ,2)) * x[k - 1]);

//Este bloco implementa o Desfazer Atualizar
x[0] -= 2 * (Round( $\delta$  * x[1]));
for (k = 2; k < comprSinal; k += 2)
    x[k] -= Round( $\delta$  * (x[k + 1] + x[k - 1]));

//Este bloco implementa o Desfazer Predizer
for (k = 1; k < comprSinal - 2; k += 2)
    x[k] -= Round( $\gamma$  * (x[k - 1] + x[k + 1]));
x[comprSinal - 1] -= Round(2 *  $\gamma$  * x[comprSinal - 2]);

//Este bloco implementa o Desfazer Atualizar
x[0] -= Round(2 *  $\beta$  * x[1]);
for (k = 2; k < comprSinal; k += 2)
    x[k] -= Round( $\beta$  * (x[k + 1] + x[k - 1]));

// //Este bloco implementa o Desfazer Predizer
for (k = 1; k < comprSinal - 2; k += 2)
    x[k] -= Round( $\alpha$  * (x[k - 1] + x[k + 1]));
x[comprSinal - 1] -= Round(2 *  $\alpha$  * x[comprSinal - 2]);
```

4.2. Implementação da DWT e da IDWT Utilizando o Esquema de *Lifting* 77

Na implementação da proposta apresentada, é usada a função pertencente ao pacote QccPack [36] que implementa esse algoritmo: `QccWAVLiftingAnalysisIntCohenDaubechiesFeauveau9_7()`. Nessa proposta, a IDWT bidimensional nível 3 é aplicada à matriz `matrix`, resultante da análise *wavelet* da imagem. De forma similar a DWT, a implementação da IDWT bidimensional aplica a transformada inversa, primeiramente, nas linhas da matriz, e, depois, nas colunas utilizando a função `QccWAVSubbandPyramidIntInverseDWT()`:

```
QccWAVSubbandPyramidIntInverseDWT(&Subband_Pyramid, &WaveletInv)
```

A variável `Subband_Pyramid`, do tipo `QccWAVSubbandPyramidInt`, contém, em seu campo `matrix`, a matriz de coeficientes organizados em uma hierarquia de sub-bandas. A variável `WaveletInv`, do tipo `QccWAVWavelet`, indica, em seu campo `implementation`, que o esquema de *lifting* é utilizado na implementação, e em seu campo `scheme` da estrutura `lifting_scheme` contém o valor que especifica que a *wavelet* a ser aplicada é a bi-ortogonal CDF 9-7 de coeficientes inteiros.

De forma similar a DWT, na IDWT, a função `QccWAVSubbandPyramidIntInverseDWT()` chama `QccWAVWaveletInverseDWT2DInt()` para calcular a IDWT no campo `matrix` de `Subband_Pyramid`, o número de níveis de decomposição da matriz é atribuído ao campo `num_levels` desta variável. Esta função chama `QccWAVWaveletSynthesis2DInt()` para cada nível de síntese, ou reconstrução.

A função `QccWAVWaveletSynthesis2DInt()` calcula a transformada inversa em um sinal bidimensional. Neste caso, o sinal é a matriz de cada nível de decomposição. Como o nível de decomposição da matriz é três, há dez sub-bandas, no primeiro nível de síntese é feita a reconstrução das sub-bandas LL_3 , HL_3 , LH_3 e HH_3 , resultando em uma única sub-banda, LL_2 .

A matriz resultante dessa operação possui nível de decomposição 2, com sete sub-bandas. Na próxima síntese, a matriz a ser reconstruída é constituída das sub-bandas LL_2 , HL_2 , LH_2 e HH_2 , o resultado da reconstrução é LL_1 . Finalmente, a matriz original é obtida com a reconstrução das sub-bandas LL_1 , HL_1 , LH_1 e HH_1 .

Para fazer a síntese do sinal bidimensional, a função `QccWAVWaveletSynthesis2DInt()` chama `QccWAVWaveletSynthesis1DInt()` para cada linha e, depois, para cada coluna da

matriz de entrada. Esta função calcula um nível de síntese em um sinal unidimensional de valores inteiros, representado por um vetor, assumindo que este contém uma sub-banda de baixa frequência em sua primeira metade e uma de alta frequência na segunda.

A função `QccWAVWaveletSynthesis1DInt()` chama `QccWAVLiftingSynthesisInt()` para calcular os passos de *lifting* no sinal. O algoritmo que implementa os passos de *lifting* que fazem a decomposição do sinal está descrito no início desta seção.

Assim a função `QccWAVWaveletSynthesis1DInt()` faz a síntese das linhas e colunas de cada sub-banda em decomposição. O resultado da síntese *wavelet* da matriz de coeficientes, é a imagem original. Em particular, nesta dissertação, é obtida a imagem mais o sinal adicionando: a marca d'água.

As próximas seções descrevem os algoritmos de construção, inserção e extração da marca d'água digital.

4.3 Construção da Marca D'água

O principal diferencial do método apresentado nesta dissertação é o seu algoritmo de construção da marca d'água, que é elaborado de forma que a marca d'água possua as seguintes propriedades: contém a assinatura digital da imagem a ser marcada, é capaz de detectar e localizar alterações na imagem marcada, é extraída sem a necessidade da imagem original, e pode ser extraída e verificada mediante o uso de informações públicas [38].

Uma característica importante do algoritmo de construção apresentado é a relação existente entre a marca d'água construída e a imagem marcada, que se deve a assinatura digital gerada. Por meio dessa relação, pode-se detectar qualquer alteração na imagem marcada ou a tentativa de inserir a marca d'água de uma imagem em outra.

Há métodos em a marca d'água é construída a partir de uma matriz pseudo-randômica gerada com uso de chaves simétricas e ou de um logotipo, de forma que não há qualquer relação entre a imagem a ser marcada e a marca d'água, [4] [28].

Outra característica importante é o uso de algoritmos que utilizam chaves públicas na construção e verificação da marca d'água. Assim, alegações de autenticidade e de integridade de imagens podem ser aceitas ou rejeitadas, sem que o proprietário precise revelar informações privadas.

Os poucos métodos que apresentam essa característica possuem algumas aspectos negativos tais como: a marca d'água não é inserida na imagem [19] [30]; e falhas no algoritmo de construção e inserção que possibilitam ataques [31], como está especificado na Seção 3.7.3.

No trabalho proposto, a marca d'água construída, denominada M , é constituída de duas imagens: (1) uma imagem significativa denominada M_S de 64x52 pixels; e (2) uma imagem não significativa de 64x8 pixels, denominada M_B . Esta imagem não significativa é construída a partir dos *bits* da assinatura digital da imagem, calculada no processo de construção da marca d'água, em função dos valores dos pixels da imagem.

Para calcular a assinatura digital, primeiramente é feita a decomposição da imagem a ser marcada por meio da DWT bidimensional nível 3. Ao decompor uma imagem utilizando uma DWT bi-dimensional nível K , obtêm-se os coeficientes *wavelet* organizados em sub-bandas de frequência: $3 \times K$ sub-bandas de detalhes, que são de frequência alta, e uma sub-banda de frequência baixa, LL_K , que é uma aproximação de baixa resolução da imagem original.

A marca d'água será inserida nos coeficientes da sub-banda LL_3 , os coeficientes das demais sub-bandas são utilizados no cálculo da assinatura digital. A Figura 4.5 apresenta um diagrama de blocos que apresenta todo o processo de construção da marca d'água M .

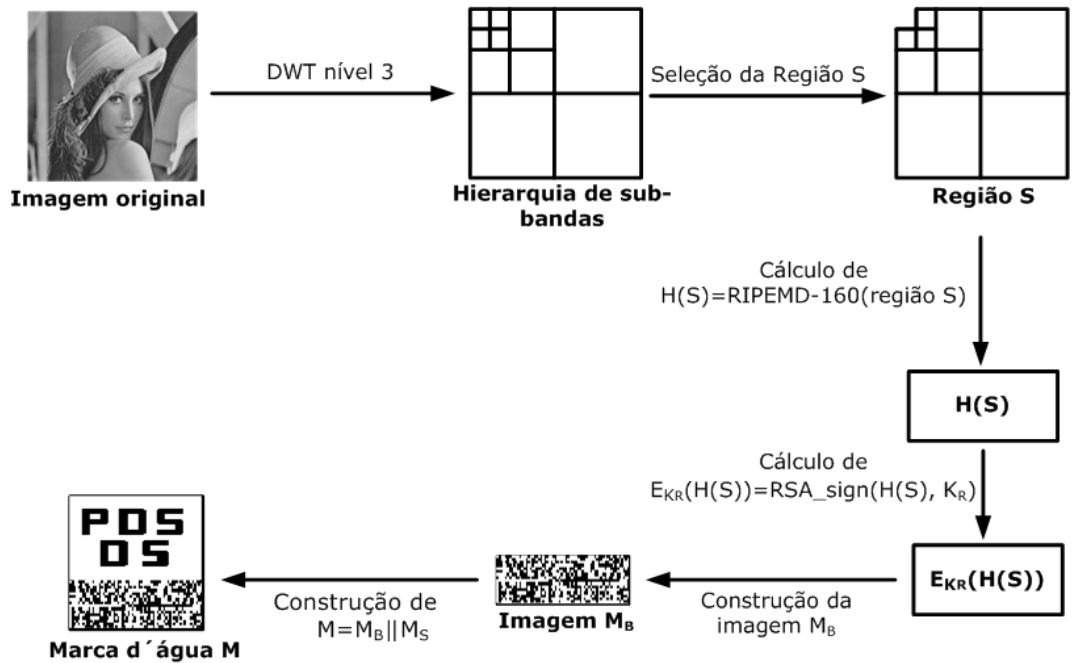


Figura 4.5: Processo de construção da marca d'água M

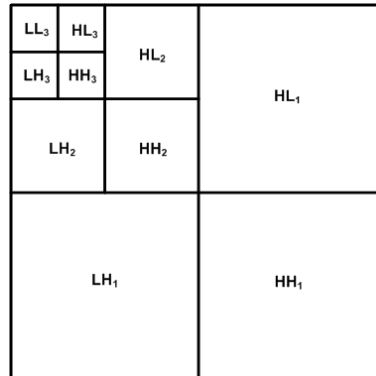
No processo de construção, primeiramente é aplicada a DWT bidimensional nível 3 à imagem a ser marcada I , por meio da função `QccWAVSubbandPyramidIntDWT()`. A seção 4.2.1 descreve o funcionamento desta função, bem como detalhes da implementação desta transformada. A Figura 4.6 mostra o resultado da decomposição da imagem Lenna, são obtidas dez sub-bandas de frequência: LH_1 , HL_1 , HH_1 , LH_2 , HL_2 , HH_2 , LH_3 , HL_3 , HH_3 e LL_3 .



(a) Imagem Lenna.



(b) Hierarquia de sub-bandas resultante da decomposição da imagem Lenna por uma DWT nível 3.



(c) esboço da hierarquia de sub-bandas de frequência.

Figura 4.6: Decomposição da imagem Lenna.

Dentre sub-bandas resultantes da decomposição, a de frequência mais baixa LL_3 é uma aproximação da imagem original em menor resolução e contém a maior parte da energia da imagem. As demais sub-bandas contêm os detalhes da imagem. A Figura 4.6(b) teve seus valores de brilho e contraste modificados para melhorar a visualização e identificação das diferentes sub-bandas de frequência, pois, originalmente, não se conseguiria visualizar os detalhes.

A marca d'água é inserida na sub-banda LL_3 , as demais sub-bandas, denominadas região S , são utilizadas para gerar a assinatura digital da imagem original. O trecho de código abaixo mostra como é construída a região S , que é carregada para a variável `regiaoS`:

```
/* Declaração da variável regiaoS */
char *regiaoS;
regiaoS = (char *)malloc(((ImageNumRows * ImageNumCols)
    - (subbandRows * subbandCols))*sizeof(char)*10);

/* Construção da regiaoS */
k=0;
for (row = 0; row < ImageNumRows; row++){
    for (col = 0; col < ImageNumCols; col++){
        if(!(row < subbandRows && col < subbandCols)){
            sprintf(saux,"%u",SubbandPyramid.matrix[row][col]);
            j=strlen(saux);
            for(i=0; i<j;i++){
                regiaoS[k]=saux[i];
                k++;
            }
            regiaoS[k]='\0';
        }
    }
}
```

Neste trecho de código, a variável `regiaoS` é um vetor de caracteres cujo tamanho é igual ao tamanho da imagem, dado por `ImageNumRows* ImageNumCols`, menos o tamanho da sub-banda LL_3 , dado por `subbandRows * subbandCols`.

A matriz `SubbandPyramid.matrix[row][col]` contém os coeficientes *wavelet* inteiros organizados em sub-bandas. Todos valores dos coeficientes desta matriz são copiados para a variável `regiaoS`, exceto os da sub-banda LL_3 , que são excluídos por meio da condição `if(!(row < subbandRows && col < subbandCols))`.

Cada coeficiente é um valor do tipo inteiro que, por meio da função `sprintf`, é convertido em uma cadeia de caracteres, ou *string*, que é armazenada na variável `saux`. Depois, cada caractere de `saux` é armazenado em `regiaoS`. O último caractere da *string* `regiaoS` é `\0` para indicar o seu final.

Após armazenar os coeficientes da região S em `regiaoS`, essa variável é passada como argumento da função *hash* `RIPEMD-160()`, para gerar o código de 160 *bits*, $H(S)$. Esse código *hash* é cifrado, utilizando a chave criptográfica privada do transmissor (proprietário)

da imagem, K_R , gerando uma assinatura digital de 512 *bits*, denominada $E_{K_R}(H(S))$.

Primeiramente, é feita a alocação da variável `hash`, que é uma *string* de 161 caracteres, destes 160 armazenam o código *hash* e o último armazena o caractere de final de *string* `\0`. A função `RIPEMD160()` calcula o código *hash* da *string* contida em `regiaoS`, cujo comprimento é `strlen(regiaoS)` e o armazena na variável `hash`, conforme mostra o trecho de código seguinte:

```
hash = (char *)malloc((RIPEMD160_DIGEST_LENGTH)*sizeof(char)+1);
hash = RIPEMD160(regiaoS, strlen(regiaoS),hash);
```

Depois, por meio da função `load_key`, a chave privada do RSA é lida de um arquivo e armazenada na variável `pkey`, do tipo `EVP_PKEY`, que é uma estrutura definida na biblioteca OpenSSL [37] para armazenar uma chave privada.

```
int formatKey=FORMAT_PEM;
char *password=NULL;
ENGINE *e = NULL;
EVP_PKEY *pkey;

pkey = load_key(bio_err,fileKey,formatKey,0, password,e,"Private Key");
if (pkey != NULL)
    rsa = EVP_PKEY_get1_RSA(pkey);
EVP_PKEY_free(pkey);
```

O primeiro parâmetro da função `load_key` é uma variável do tipo `BIO`, `bio_err`, caso ocorram erros enquanto a chave é carregada, as mensagens correspondentes são armazenadas nesta variável. Há casos de erro, especificamente os mais graves, em que todas as mensagens armazenadas nessa variável são impressas.

O segundo parâmetro, `fileKey`, é uma *string* que contém o caminho completo para o arquivo que armazena a chave privada. O terceiro parâmetro contém o valor da constante `FORMAT_PEM`, que especifica o formato em que a chave é armazenada. Nesse caso, a chave está armazenada no formato PEM, padrão do OpenSSL, mas também são possíveis outros formatos, especificamente DER e NET. O quarto parâmetro pode assumir valor 0 ou 1 para indicar que a chave privada deve ser lida de um arquivo ou da entrada padrão, respectivamente.

É possível proteger a chave privada utilizando uma senha que deve ser informada quando a chave é utilizada. Neste caso, o quinto parâmetro, `password`, deve conter esta

senha, caso contrário deve ser nulo.

Há casos em que são necessárias funções especiais para tratamento das chaves do RSA, por exemplo quando a chave é armazenada em algum *hardware* de forma embarcada. Nesses casos, o sexto parâmetro, a variável **e**, do tipo **ENGINE**, especifica quais funções o OpenSSL deve utilizar para efetuar as operações do RSA. Nesta implementação são usadas as funções do método padrão do OpenSSL, para tal este parâmetro deve ser nulo.

O sétimo parâmetro, a *string* "Private Key", indica que a chave a ser carregada é a secreta. Se a chave é carregada com sucesso para **pkey**, a função **EVP_PKEY_get1_RSA()** a carrega para a variável **rsa** do tipo **RSA**. Esta estrutura armazena os elementos que constituem as chaves do RSA, seus principais campos são:

```
struct{
    BIGNUM *n;           // public modulus
    BIGNUM *e;           // public exponent
    BIGNUM *d;           // private exponent
    BIGNUM *p;           // secret prime factor
    BIGNUM *q;           // secret prime factor
    // ...
}; RSA
```

As chaves, pública e privada, do RSA são construídas a partir de dois números primos muito grandes, que são armazenados nos campos **p** e **q** da estrutura **RSA**. O campo **n**, resultante do produto de **p** e **q**, é utilizado para cifrar e decifrar mensagens. O campo **e** é um número pseudo aleatório que deve satisfazer as condições: $e < \phi(n) = (p-1)*(q-1)$ e $\text{mdc}(e, \phi(n)) = 1$. O campo **d** é calculado a partir de **e** da seguinte forma: $e*d \equiv 1 \text{ mod } \phi(n)$ ³. A chave pública é constituída de **e** e de **n** e a privada dos demais números, **p**, **q**, e **d**.

Uma vez que a chave privada é carregada para a variável **rsa**, finalmente, a assinatura digital é calculada por meio da função **RSA_sign()**:

```
char *sigret;
sigret = (char *) (malloc(RSA_size(rsa))+1);
RSA_sign(NID_ripemd160,hash,RIPEMD160_DIGEST_LENGTH,sigret,&siglen,rsa);
```

Essa função assina a mensagem armazenada na variável **hash**, que contém o código **hash**, usando a chave privada armazenada na variável **rsa**. A assinatura digital gerada,

³Nesta equação, o operador $\equiv$ representa a relação de congruência, sendo que $a \equiv b \text{ mod } n \Leftrightarrow n$ divide $(a - b) \Leftrightarrow \exists c; a - b = c * n \Leftrightarrow a = c * n + b$. Por exemplo, se $a \equiv 3 \text{ mod } 7$, então $a = 3, 10, 17, 24, \dots$

cujo tamanho é armazenado em `siglen`, é retornada na variável `sigret`. Os parâmetros `NID_ripemd160` e `RIPEMD160_DIGEST_LENGTH`, definidos na biblioteca OpenSSL, informam qual o algoritmo utilizado para gerar o código `hash` e qual o tamanho deste código, respectivamente.

A assinatura digital resultante é interpretada como uma imagem, cada *bit* da assinatura digital é mapeado em um pixel de uma imagem binária não significativa denominada M_B , de 64x8 pixels. Essa imagem, junto com a imagem significativa M_S , de 64x52 pixels, forma a marca d'água M , de 64x64 pixels, a ser inserida em I . Sintaticamente, cada *bit* da assinatura digital é inserido na imagem conforme descreve a seção 4.4.

A Figura 4.7 mostra as duas imagens que constituem a marca d'água: a imagem significativa, M_S , e a não significativa, M_B . A imagem M_S está armazenada no formato PBM e é constituída de pixels de valor igual a 0 ou 1, que resultam na cor preta e branca, respectivamente. Essa imagem é fixa, ou seja, faz parte da marca d'água a ser inserida em todas as imagens. Já M_B é única para cada imagem, pois é a sua assinatura digital.



(a) Imagem significativa M_S .



(b) Imagem binária não significativa M_{BS} .

Figura 4.7: Imagens que constituem a marca d'água.

Essa marca d'água é inserida na imagem por meio do método de inserção apresentado na próxima seção.

4.4 Inserção da Marca D'água nos Coeficientes *Wavelet*

Atualmente, existem muitos algoritmos de inserção de marca d'água baseados na transformada *wavelet*. Um dos métodos pioneiros e bastante conhecido é o de Cox, em que a marca d'água é inserida nos coeficientes *wavelet*, levando em consideração a sua energia: a robustez da marca d'água é proporcional à capacidade perceptiva e à grandeza do coeficiente [23].

No algoritmo apresentado nesta dissertação, a transformada *wavelet* inteira é aplicada à imagem original para que se possa identificar as regiões onde a marca d'água pode ser inserida de forma que não provoque distorções visíveis ao HVS. O algoritmo que implementa esta transformada está descrito na seção 4.2.

A inserção da marca d'água é feita nos *bits* menos significativos dos coeficientes da sub-banda LL_3 mediante o uso de equações de inserção apresentadas nesta seção. Há outros algoritmos de inserção que inserem a marca d'água por meio de métodos de quantização que geram um novo valor para o coeficiente [24] [25]. A Figura 4.8 apresenta um diagrama de blocos com os principais elementos e com os passos do processo de inserção da marca d'água.

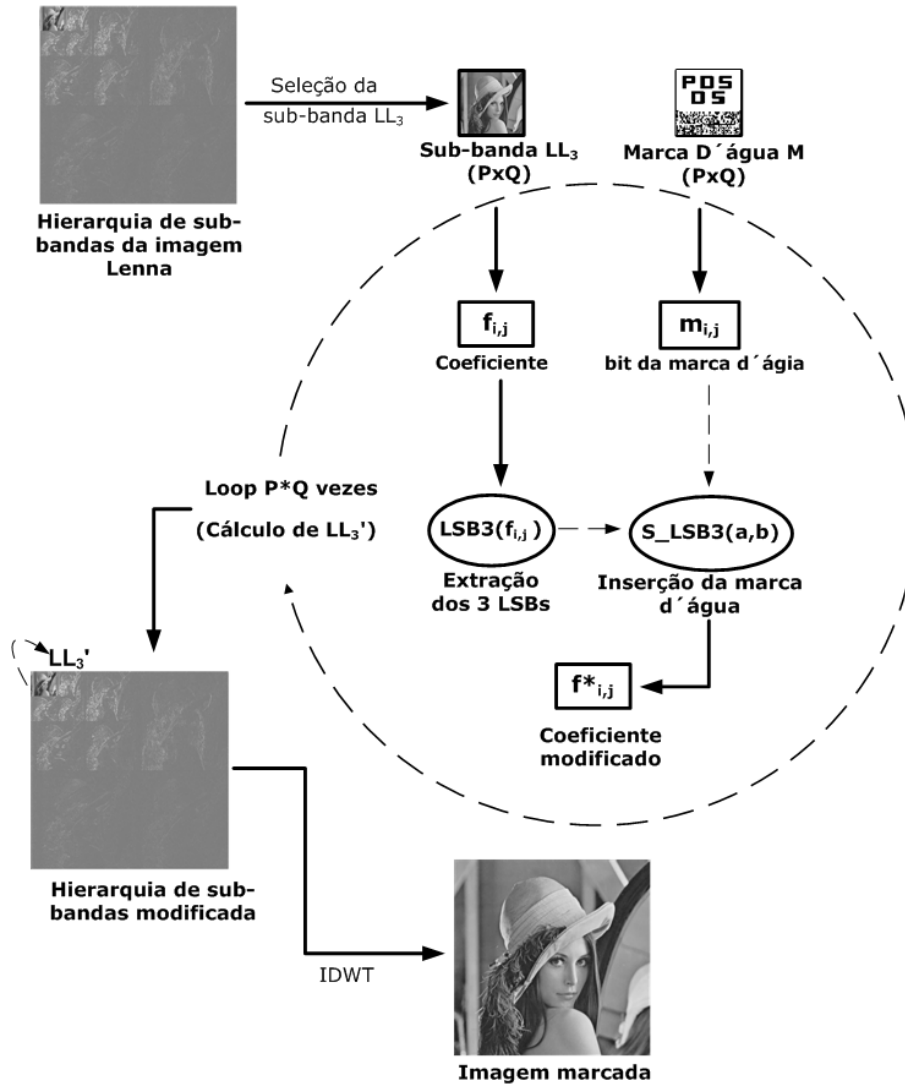


Figura 4.8: Processo de inserção da marca d'água M na imagem I

Para que esteja de acordo com o HVS, o algoritmo de inserção de marca d'água deve levar em consideração que a visão humana é menos sensível ao ruído nas seguintes regiões da imagem: (1) onde o brilho é muito intenso ou muito baixo; e (2) onde a textura é forte, especialmente próximo aos contornos, ou descontinuidades. Essas regiões da imagem são isoladas na sub-banda de baixa frequência, LL_K , resultante da decomposição *wavelet* [28].

As sub-bandas de frequência resultantes da decomposição de uma imagem por uma DWT possuem a seguinte propriedade: quanto menor a escala da sub-banda, menor é a sua energia e maior é a sua resolução e frequência. Além disso, os valores dos coeficientes *wavelets* são, em média, maiores nas sub-bandas de menor energia, e quanto maior o valor, maior a importância visual do coeficiente. Com isso, tem-se que os coeficientes da sub-banda LL_K possuem algumas características específicas não encontradas nas demais sub-bandas, tais como [39]:

- Contêm a maior parte da energia e formam uma aproximação da imagem original;
- São bem preservados quando a imagem passa por alguns tipos de processamento, que na maioria das vezes alteram mais os componentes de frequência alta, por exemplo compressão utilizando tecnologia JPEG;
- Possui alta capacidade perceptual, o que garante que a inserção de uma marca d'água de robustez razoável não provoque distorções perceptíveis na imagem marcada.

Essas características da sub-banda LL_K estão de acordo com os itens que descrevem as regiões da imagem onde a visão humana é menos sensível ao ruído. Por causa de todas essas propriedades, o método de inserção aplica a DWT bidimensional nível 3 à imagem e insere a marca d'água na sub-banda LL_3 . O nível 3 de decomposição foi escolhido para que os coeficientes da sub-banda de inserção tenham energia suficiente para manter a marca d'água invisível e para que ela tenha tamanho razoável, de modo que a detecção de adulteração ofereça informações mais precisas na localização das regiões adulteradas.

Se I é uma imagem de tamanho $M \times N$, a dimensão da sub-banda de frequência LL_K , denominada $P \times Q$, é igual a $\frac{1}{2^K} \times M \times N$, conforme está detalhado na Seção 4.2.1. No algoritmo de inserção apresentado nesta dissertação, tem-se que $K = 3$, a marca d'água tem tamanho igual ao da sub-banda LL_3 e a inserção é feita mediante a substituição dos *bits* menos significativos dos coeficientes desta sub-banda [28].

O método proposto, além do algoritmo de construção da marca d'água, possui outro diferencial em relação aos demais publicados. No algoritmo de inserção original, apresentado no Capítulo 3, o número de *bits* substituídos é cinco, para garantir invisibilidade da marca d'água enquanto mantém certo nível de robustez, e os números binários 11000_2 e 01000_2 representam os *bits* 1 e 0, respectivamente, para maximizar a robustez [28].

A marca d'água apresentada nesta dissertação é frágil, por isto o número de *bits* substituídos é reduzido para três e os números binários 110_2 e 010_2 representam os *bits* 1 e 0, respectivamente. Essa medida possibilita reduzir a distorção provocada na imagem resultante da inserção da marca d'água.

Dentre as duas imagens que constituem a marca d'água, M_S está armazenada na variável **sigret**, calculada no processo de construção, já M_B , está armazenada em um arquivo no formato PBM. Antes de fazer a inserção, essa imagem é carregada para a variável **Watermark**, do tipo **QccIMGImage**, chamando a função **QccIMGImageRead(&Watermark)**, especificada na seção 4.2.

As equações 4.16 e 4.17 são utilizadas para inserir os *bits* 0 e 1 da marca d'água M na imagem I , respectivamente. A substituição dos *bits* é feita de tal forma que a invisibilidade da marca d'água seja garantida. Nessas equações, a função $S_LSB3(a, b)$ substitui os 3 *bits* menos significativos de a por b , a função $LSB3(a)$ retorna os 3 bits menos significativos de a , m_{ij} representa um *bit* da marca d'água M a ser inserido, f_{ij} representa um coeficiente da sub-banda LL_3 antes da inserção de m_{ij} , e f_{ij}^* representa este coeficiente após da inserção de m_{ij} , onde $0 \leq i \leq P$ e $0 \leq j \leq Q$.

Quando o *bit* $m_{ij} = 1$, a seguinte equação de inserção é utilizada:

$$f_{ij}^* = \begin{cases} S_LSB3(f_{ij} - 010_2, 110_2), & LSB3(f_{ij}) \leq 010_2 \\ S_LSB3(f_{ij}, 110_2) \end{cases} \quad (4.16)$$

Quando $m_{ij} = 0$, a seguinte equação de inserção é utilizada:

$$f_{ij}^* = \begin{cases} S_LSB3(f_{ij} + 100_2, 010_2), & LSB3(f_{ij}) \geq 110_2 \\ S_LSB3(f_{ij}, 010_2) \end{cases} \quad (4.17)$$

O PSNR mede a similaridade entre uma imagem original I , de tamanho $M \times N$, e uma imagem I' , resultante de um processamento aplicado a I , com possível adição de ruído.

Quanto maior o seu valor, maior a similaridade das imagens. O PSNR, em decibéis (dB), é calculado por [40]:

$$PSNR = 20 * \log_{10} \left(\frac{255}{\sqrt{MSE}} \right), \quad (4.18)$$

onde

$$MSE = \frac{\sum [I(i, j) - I'(i, j)]^2}{M * N}. \quad (4.19)$$

Com esse método de inserção, para todas as instâncias, a diferença entre o valor do coeficiente antes e após a inserção, denominada Δ , fica entre -3 +4. Ou seja, essas equações reduzem o intervalo de variação da magnitude do coeficiente DWT resultante da inserção do *bit* da marca d'água.

No método original o Δ fica entre -15 +16. Com a modificação, os resultados experimentais mostram que o PSNR obtido é melhor que aquele obtido com o algoritmo de inserção original em aproximadamente 3 dB [28]. O trecho de código a seguir mostra como as Equações 4.16 e 4.17 são implementadas em linguagem C:

```
resultI = coeficiente_pixel & 0x07;
/* 0 bit da marca d'água vale 0: implementação da Equação 4.17 */
if(resultW){
    if(resultI <= 2){
        y = coeficiente - 2;
        coeficiente = ((y & ~0x07) | 6);
    }
    else{
        y = coeficiente;
        coeficiente = ((y & ~0x07) | 6);
    }
}
/* 0 bit da marca d'água vale 0: implementação da Equação 4.16 */
else{
    if(resultI >= 6){
        y = coeficiente + 4;
        coeficiente = ((y & ~0x07) | 2);
    }
    else{
        y = coeficiente;
        coeficiente = ((y & ~0x07) | 2);
    }
}
```

Primeiramente, os três LSBs do coeficiente *wavelet* em que os *bits* da marca d'água será inserido são obtidos: o operador $\&$ efetua a operação lógica *E* entre cada *bit* dos

operandos `coeficiente` e `0x07` (decimal sete representado em hexadecimal). O resultado desta operação é armazenado na variável inteira `resultI`⁴.

A variável inteira `resultW` armazena um pixel da marca d'água, se `resultW` tem valor um, tem-se a implementação da Equação 4.17: se `resultI` tem valor menor ou igual a dois, então o valor 2 é subtraído do valor do coeficiente *wavelet*, e seus três LSBs são substituídos pelo valor seis, caso contrário, é feita apenas a substituição. A implementação da Equação 4.16 é feita de forma similar, alterando apenas o valor subtraído e o da substituição.

A substituição é feita utilizando os operadores `&`, `|` e `~`, sendo que o segundo operador efetua a operação lógica *OU* entre cada *bit* de seus argumentos e o último operador inverte cada *bit* de seu argumento, por exemplo `~00001111=11110000`. Ao calcular `y&~0x07`, os três LSBs de `y` são zerados, depois são substituídos por `0x110`, representação de seis em hexadecimal, por meio do operador `|`, pois `a|0=a`.

Após a inserção dos *bits* de *M* na imagem *I*, a IDWT bidimensional nível 3 é aplicada nos coeficientes da imagem por meio da função `QccWAVSubbandPyramidIntInverseDWT()` e a imagem marcada *I'* é obtida, sendo visivelmente idêntica a *I*. A seção 4.2.2 descreve o funcionamento desta função, bem como detalhes da implementação desta transformada.

Este método de inserção garante a ausência de distorções perceptíveis ao HVS e valores altos de PSNR. A próxima seção mostra como é feita a extração da marca d'água e a verificação da assinatura digital.

4.5 Extração e Verificação da Marca D'água

Após a transmissão da imagem marcada *I'*, a verificação da marca d'água é feita no lado receptor, onde a imagem passa a ser chamada *I''*, uma vez que, durante a sua transmissão, é possível que *I'* sofra adulterações.

No lado receptor, para extrair a marca d'água *M'*, a DWT bidimensional nível 3 é aplicada na imagem possivelmente marcada *I''*, obtendo os coeficientes *wavelet*. A marca d'água é extraída dos coeficientes da sub-banda *LL₃* utilizando o algoritmo de extração descrito nesta seção.

⁴O operador lógico `&`, por definição, possui as seguintes propriedades: `a&0 = 0` e `a&1 = a`. Considerando que a variável coeficiente possui 32 *bits*, o resultado da operação `y&0x07`, armazenado em `resultI`, possui 29 zeros em seus MSBs, e os três LSBs de coeficiente

A Figura 4.9 apresenta o diagrama de blocos do processo de extração da marca d'água. Uma característica importante a ser observada é que esta é uma marca d'água do tipo *blind* ou *oblivious*, pois a imagem original não é necessária durante o processo de extração da marca d'água.

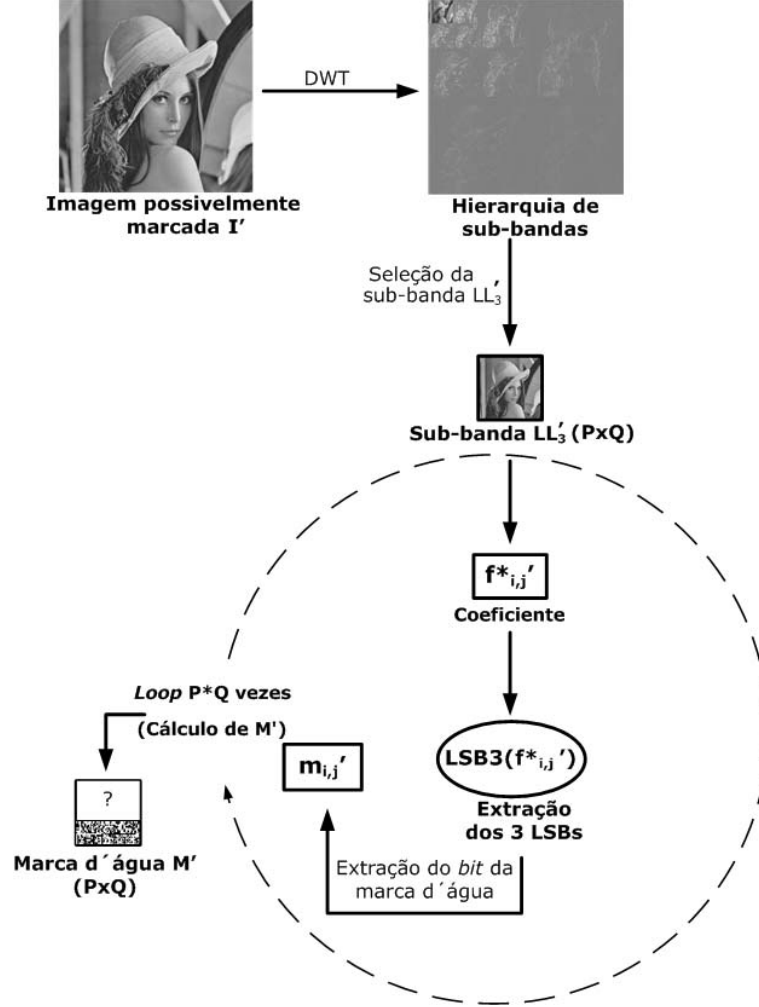


Figura 4.9: Processo de extração da marca d'água M da imagem I

Para extrair a marca d'água M' , a DWT bidimensional nível 3 é aplicada na imagem possivelmente marcada I' , obtendo os coeficientes *wavelet* $f_{ij}^{*'}$. Seja m_{ij}' o *bit* da marca d'água extraída e $LSB3(a)$ a função que retorna os 3 *bits* menos significativos de a , a Equação 4.20 é utilizada para obter m_{ij}' , onde $0 \leq i \leq P$ e $0 \leq j \leq Q$

$$m_{ij}' = \begin{cases} 1, & LSB3(f_{ij}^{*'}) = 0 \\ 0, & LSB3(f_{ij}^{*'}) = 1 \end{cases} \quad (4.20)$$

Esta fórmula é aplicada a todos os coeficientes de LL_3'' , resultantes da decomposição por meio da DWT, até que todos os bits m_{ij}' sejam obtidos, estes *bits* irão representar os pixels da marca d'água extraída M' .

O trecho de código a seguir mostra como a Equação 4.20 é implementada em linguagem C:

```
resultI = coeficiente_pixel & 0x07;
result0 = resultI ^ 2; /*Se LSB = 2 (representação do bit 1) */
if(!result0)
    watermark_pixel= 0;
else
    watermark_pixel= 1;
```

De forma similar a implementação do método de inserção, os LSBs do coeficiente *wavelet* marcado é obtido por meio do operador $\&$, o resultado é armazenado na variável de tipo `int resultI`. A variável `result0` armazena o resultado da operação lógica $\oplus$ entre cada *bit* dos argumentos deste operador: `resultI` e o inteiro 2^5 .

Assim, quando `resultI` vale dois, então `result0` vale zero, isto implica que `!result0` é verdadeiro, logo, o pixel da marca d'água, `watermark_pixel`, deve ter valor zero. Caso contrário, quando `resultI` tem valor seis, então `result0` vale quatro ($0x100 = 0x110 \wedge 0x010$), logo, o pixel da marca d'água, `watermark_pixel`, deve ser um. Em casos em que há erro na marca d'água, o pixel recebe valor um.

A marca d'água extraída, M' , é constituída da imagem não significativa M'_B , e da imagem significativa M'_S . A primeira é utilizada como parâmetro durante o processo de verificação da assinatura digital, apresentado na Seção 4.5.1. E a segunda é comparada com a imagem significativa original, M_S , para localizar as regiões adulteradas. Os parágrafos seguintes especificam como

O algoritmo de extração calcula o valor do *bit* da marca d'água que está inserido no coeficiente *wavelet*. A assinatura digital é uma cadeia de caracteres, sendo que cada caractere possui oito *bits*. Assim, é necessário juntar os *bits* calculados na extração para que formem um caractere para reconstruir a assinatura digital. O trecho de código abaixo mostra os detalhes da implementação:

⁵Na operação $\oplus$, ou *OU – exclusivo*, quando os dois argumentos são iguais o resultado é zero, caso contrário é um. Assim, o resultado da operação $\wedge$ será uma seqüência de zeros quando os dois argumentos forem iguais

```

char bitVec[8] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
assinatura_char = 0x00;

for(j=0; j<8; j++){
    aux = replace_bits_x1(subband_pyramid->matrix[row][col]);
    if(aux & 0x01)
        assinatura_char = assinatura_char ^ bitVec[j];

assinatura[i] = assinatura_char;
assinatura_char = 0x00;

```

No código supramencionado, a variável `assinatura_char` armazena um caractere, esta variável é inicializada com zero. A função `replace_bits_x1()` extrai o *bit* da marca d'água⁶. Caso o *bit* extraído tenha valor 1, então este valor deve ser atribuído ao *bit* da posição corrente do caractere armazenado em `assinatura_char`.

Essa atribuição é feita mediante o operador `^`. Por exemplo, se `assinatura_char` correntemente armazena a seqüência 01101000, a posição corrente é a 5, e o *bit* tem valor 1. Neste caso, será feita a operação $00000110 \wedge 00100000$ cujo resultado é 0010011, em que o *bit* da posição 5 recebe valor 1. Assim, o vetor `bitVec` é construído de forma que o *bit* da posição corrente seja “ligado” caso o *bit* da marca d'água seja 1.

O caractere calculado em cada passo é armazenado em uma posição de `assinatura`, uma *string* que, no final, conterá a assinatura digital. Ao final do cálculo de cada caractere, a variável `assinatura_char` deve ser inicializada novamente. A próxima seção descreve como é feita a verificação da assinatura digital.

4.5.1 Verificação da Assinatura Digital

Durante o processo de extração, a DWT é aplicada na imagem I'' e os 512 *bits* da assinatura digital, $E_{K_R}(H(S))'$, são obtidos a partir da imagem M'_B extraída da sub-banda LL_3'' , conforme está apresentado na seção 4.5. Essa assinatura digital é decifrada utilizando a chave pública do transmissor para a obtenção do código *hash*.

Além disso, um novo código *hash* é calculado em função dos coeficientes da região S da imagem recebida, de forma similar ao cálculo feito processo de inserção. Os dois códigos obtidos são comparados, caso sejam iguais, a imagem é íntegra e autêntica. A Figura 4.10

⁶A função `replace_bits_x1()` implementa o código de extração da marca d'água descrito anteriormente.

ilustra o processo de verificação da marca d'água.

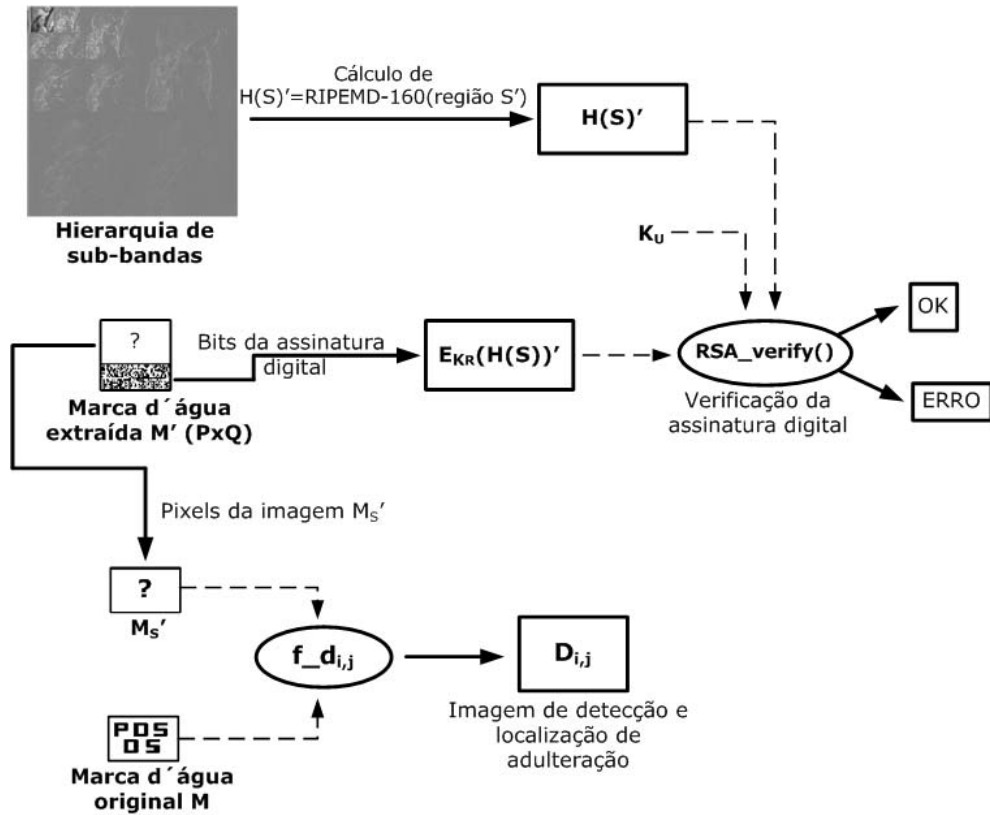


Figura 4.10: Processo de verificação da assinatura digital

Assim como no processo de construção da marca d'água, apresentado na seção 4.3, um código *hash* da região S' , $H(S)'$, é calculado. Esse código, juntamente com a assinatura digital extraída e a chave pública do proprietário da imagem (K_P), $E_{K_R}(H(S))'$, é passado como parâmetro para a função `RSA_verify()`, que faz a verificação da assinatura.

A implementação da verificação da assinatura digital é similar à implementação do cálculo da assinatura digital, apresentado na seção 4.3. Os primeiros passos da verificação da assinatura digital são o armazenamento da região S' na variável `regiaoS` e o cálculo do código *hash* $H(S)'$. Assim como no algoritmo de construção da marca d'água, isto é feito por meio da função `RIPEMD-160()`, só que neste caso, `regiaoS` contém a região S' .

Depois, por meio da função `load_pubkey`, a chave pública do RSA é lida de um arquivo e armazenada na variável `pkey`, do tipo `EVP_PKEY`, que é uma estrutura definida na biblioteca OpenSSL [37] para armazenar uma chave privada.

```
int formatKey=FORMAT_PEM;
char *password=NULL;
```

```
ENGINE *e = NULL;
EVP_PKEY *pkey;

pkey = load_pubkey(bio_err, fileKey, formatKey, 1, password, e, "Public Key");
if (pkey != NULL)
    rsa = EVP_PKEY_get1_RSA(pkey);
EVP_PKEY_free(pkey);
```

Os parâmetros da função `load_pubkey` são semelhantes aos da função `load_key`, descrita na seção 4.3, exceto o último parâmetro, "PublicKey", que especifica que a chave a ser carregada é pública. Se a chave é carregada com sucesso para `pkey`, a função `EVP_PKEY_get1_RSA()` a carrega para a variável `rsa`, do tipo `RSA`.

Uma vez que a chave pública é carregada para a variável `rsa`, tem-se todas as informações necessárias para fazer a verificação da assinatura digital por meio da função `RSA_verify()`:

```
RSA_verify(NID_ripemd160, hash, RIPEMD160_DIGEST_LENGTH, assinatura, siglen, rsa)
```

Essa função verifica se a assinatura armazenada em `assinatura`, de tamanho `siglen`, corresponde ao código armazenado na variável `hash`, que contém o código *hash* de I'' . No processo de verificação, a assinatura digital é decifrada utilizando a chave pública, armazenada em `rsa`, o resultado é o código *hash* $H(S)'' = D_{K_U}(E_{K_R}(H(S))')$. Os códigos $H(S)'$ e $H(S)''$ são comparados para validar a assinatura.

Caso os códigos sejam iguais, tem-se a garantia de que a imagem não sofreu qualquer tipo de adulteração e de que a origem alegada é realmente o transmissor (ou proprietário). Caso contrário, há duas possibilidades: a imagem foi adulterada ou não foi marcada pela origem proprietária das chaves privada K_R e pública K_U . Nesses dois casos, a imagem não é autêntica.

A verificação da assinatura digital possibilita a detecção de adulterações na imagem, bem como a legitimidade de sua origem. A próxima seção apresenta o método de localização de adulterações.

4.5.2 Localização de Adulterações

Se na verificação da assinatura digital é determinado que a imagem é autêntica, não haverá regiões adulteradas. Caso contrário, a marca d'água é extraída com erro e estas regiões

podem ser localizadas por meio de uma imagem denominada mapa de localização de adulterações, D_{ij} , que é calculada de acordo com a Equação 4.21, onde $0 \leq i \leq P$ e $0 \leq j \leq Q$

$$D_{ij} = |m_{ij} - m'_{ij}|, \quad D_{ij} \in 0, 1 \quad (4.21)$$

Se $D_{ij} = 1$, então o pixel do mapa de localização é branco e indica que a marca d'água foi corretamente extraída nesta posição, caso contrário o pixel é preto e representa erro de extração. Estes pontos do mapa representam a região adulterada na imagem com uma proporção de 1 x 8 pixels. Esta proporção deve-se à diferença de tamanho entre a imagem e a marca d'água. O trecho de código abaixo mostra como é implementado o cálculo do mapa:

```
if(Watermark.Y.image[row][col] != OutputWatermark.Y.image[row][col]){
    DetectionImage.Y.image[row][col] = 0;
    tamper++;
}
else
    DetectionImage.Y.image[row][col] = 1;
```

Ao invés de calcular a diferença entre os pixels das marcas d'água, estes são apenas comparados, tal medida evita o resultado indesejável -1. Caso os pixels sejam diferentes, então houve erro na extração. Assim, caso haja erro durante a verificação de assinatura, as regiões adulteradas podem ser localizadas por meio do mapa de localização.

4.6 Experimentos e Resultados Obtidos

Esta seção apresenta os resultados de alguns experimentos e aponta os principais aspectos de segurança e complexidade computacional do método apresentado.

Os experimentos realizados visam a testar a invisibilidade da marca d'água, a sua fragilidade e a sua capacidade de detectar e de localizar adulterações. Os testes também visam a comprovar a *performance* computacional do método, por isto são apresentados os tempos gastos na inserção e na extração da marca d'água em algumas imagens.

Todos os testes foram realizados no sistema computacional cujas características estão descritas na seção 4.1. Durante a realização de todos os testes alguns programas permaneceram em execução, mantendo a mesma carga de processamento da máquina. Estes pro-

gramas são os seguintes: ambiente gráfico de trabalho KDE [41], o terminal X, Konsole [41], o ambiente LaTeX integrado ao KDE, Kile [42], o editor de textos avançado, Kate [43], e o visualizador de arquivos PostScript e PDF (Formato de Documento Portável), Kghostview [41].

4.6.1 Qualidade da Imagem Marcada e *Performance* Computacional

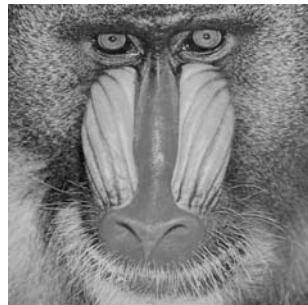
A Figura 4.11 mostra a invisibilidade da marca d'água, após a sua inserção nas imagens originais das figuras 4.11(a) e 4.11(c), ambas de tamanho 512 x 512 pixels. Doravante, as imagens dessas figuras serão referenciadas como Lenna e Baboon, respectivamente. A marca d'água inserida nas imagens é apresentada na figura 4.7 da seção 4.3.



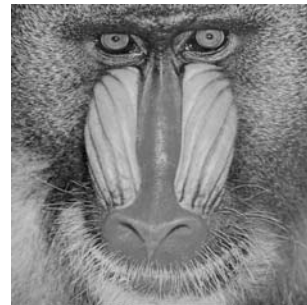
(a) Imagem Lenna original



(b) Imagem Lenna marcada



(c) Imagem Baboon original



(d) Imagem Baboon marcada

Figura 4.11: Invisibilidade da marca d'água.

A Tabela 4.1 mostra os resultados, de tempo e qualidade das imagens marcadas, obtidos com a inserção e extração da marca d'água em 21 imagens, mostradas no Apêndice A, que normalmente são utilizadas para testar esquemas de compressão de imagens e de marca d'água.

Nestes resultados, nota-se que o tempo gasto na inserção é sempre maior que o gasto na extração da marca d'água. Esta diferença deve-se ao fato de que, na inserção é necessário

Tabela 4.1: Resultados

<i>Imagem</i>	<i>Tempo de inserção (seg.)</i>	<i>Tempo de extração (seg.)</i>	<i>PSNR (dB)</i>
Avião	0.804982	0.562512	45.983791
Baboon	0.808955	0.579392	46.034644
Badlands	0.783943	0.510445	45.915266
Barbara	0.812140	0.532968	46.066527
Coroa	0.960181	0.529982	46.053241
Elaine	1.049940	0.531724	46.009026
F16	0.898360	0.514502	45.891134
Goldhill	0.823837	0.526546	46.025391
House	0.970968	0.531960	45.918659
Lago	0.966121	0.554434	46.002212
Lenna	0.786980	0.526870	46.055963
Peppers	0.802950	0.562988	46.138716
Ponte	0.962144	0.575213	45.997211
Tanque1	0.974369	0.538067	46.106570
Tanque2	0.974095	0.533961	46.043558
Tanque3	0.952060	0.550240	46.106404
Tanque4	0.975515	0.537773	46.045772
Tanques	0.962370	0.532543	46.011294
Tiffany	0.966253	0.562839	46.036567
Truck	1.006340	0.561441	46.058439

aplicar a IDWT para obter a imagem marcada, enquanto no processo de extração isto não é necessário.

Não há registros na literatura de resultados de marcas de tempo para inserção e extração de marca d'água em imagens digitais para que se faça comparações. No entanto, os métodos de inserção de marca d'água baseados na DCT requerem maior complexidade computacional do que os baseados na DWT. Dado um sinal de comprimento n , a complexidade computacional do cálculo da DWT e da DCT aplicadas ao sinal são da ordem de $O(n)$ e $O(n \log(n))$, respectivamente [4].

O esquema *lifting* é um método eficiente para melhorar a velocidade de processamento da DWT, embora mantenha a complexidade $O(N)$, esse esquema reduz, pela metade, o número de operações necessárias para o cálculo de um passo de transformada, resultando na redução pela metade da complexidade computacional comparado ao esquema clássico via banco de filtros [7] [16]. Além disto, as variáveis que armazenam os valores dos coeficientes envolvidos no cálculo de um passo da DWT podem ser sobrescritas [15], o que o

faz mais eficiente em termos de espaço também.

Portanto, a proposta apresentada nesta dissertação tem complexidade $O(N)$, pois utiliza a DWT, e a sua complexidade computacional é reduzida pela metade comparada aos métodos propostos que utilizam banco de filtros ao invés do esquema de *lifting* para implementar a DWT. Dentre os trabalhos citados na Seção 3.7.3, alguns utilizam a DWT:

1. *A review of watermarking and the importance of perceptual modeling.*
2. *Wavelet-Based Semi-Fragile Watermarking with Tamper Detection.*
3. *A secure semi-fragile watermarking for image authentication based on integer wavelet transform with parameters.*
4. *Dual Domain Watermarking for Authentication and Compression of Cultural Heritage Images.*
5. *A Robust DWT-Based Blind Data Hiding Algorithm.*

Dentre esses trabalhos, apenas o 3 utiliza o esquema de *lifting* em sua implementação, e o trabalho 4 utiliza a DCT junto com a DWT [23] [24] [4] [25] [28]. Conseqüentemente, todos os trabalhos exceto o 3 apresentam maior complexidade computacional na implementação da DWT. A análise de outros aspectos, tais como a complexidade do algoritmo de construção, inserção e extração da marca d'água desses trabalhos para comparar com a proposta apresentada foge ao escopo desta dissertação.

Além disso, os resultados apresentados mostram que a qualidade visual da imagem marcada é superior à de outros métodos *blind* que utilizam a DWT. Por exemplo, em outras publicações, a imagem Lenna marcada possui o PSNR igual a 44.8 dB [28] e 42.26 dB [4]. E a imagem Peppers marcada possui o PSNR igual a 35.91 dB [44], e 41.6 dB [24]. Comparando com esses exemplos, em todos os casos o PSNR obtido com a proposta apresentada é maior, o significa que há maior similaridade entre a imagem original e a marcada.

4.6.2 Aspectos de Segurança, Detecção e Localização de Adulterações

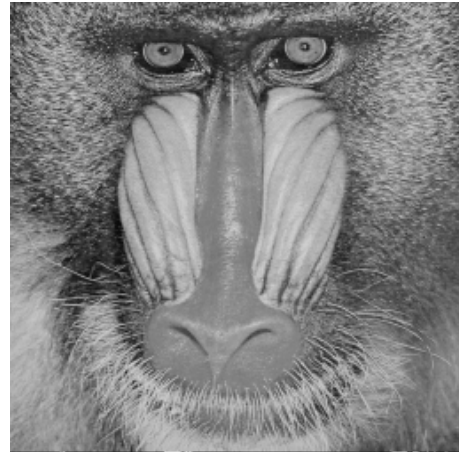
Esta seção apresenta os resultados obtidos com a simulação de alguns ataques à imagens marcadas.

Modificação Casual da Imagem Marcada

Este é o tipo de ataque mais simples, a imagem é modificada como se não houvesse marca d'água. Foram feitos testes com as imagens marcadas Lenna e Baboon. A imagem Lenna foi adulterada com a adição de um terceiro olho, conforme mostra a Figura 4.12(a).



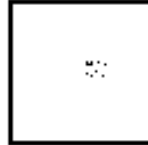
(a) Imagem Lenna adulterada



(b) Imagem Baboon adulterada



(c) Marca d'água extraída de Lenna



(d) Mapa de localização de adulterações de Lenna



(e) Marca d'água extraída de Baboon



(f) Mapa de localização de adulterações de Baboon

Figura 4.12: Fragilidade da marca d'água e capacidade de detectar as regiões adulteradas.

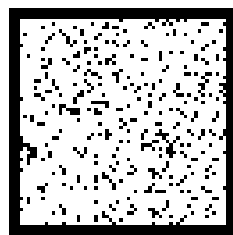
A marca d'água é extraída com erro, como mostra a Figura 4.12(c), os pontos que não foram extraídos com sucesso indicam, proporcionalmente, a localização da adulteração na imagem marcada. Os pixels pretos que aparecem no mapa de detecção, Figura 4.12(d), indicam os pontos da marca d'água extraídos com erro. Estes pontos representam a região adulterada na imagem com uma proporção de 1 x 8 pixels, por causa da diferença de tamanho entre a imagem e a marca d'água.

A imagem Baboon foi alterada com o aumento do brilho em seu nariz. A Figura 4.12(b) mostra a imagem adulterada, 4.12(e) a marca d'água extraída com erro, e 4.12(f) o mapa de detecção com a localização da região adulterada. Neste mapa, os pixels pretos indicam erro na extração da marca d'água.

Além de detectar e localizar as regiões alteradas, nos testes com as duas imagens o processo de extração da marca d'água indicou erro na verificação da assinatura digital.

Modificação do Tamanho da Imagem Marcada

Mudanças no tamanho de imagens também resultam em falha na extração da marca d'água. Neste teste, o tamanho da imagem Lenna é alterado de 512x512 pixels para 505x505 pixels. A Figura 4.13(a) mostra a marca d'água extraída e a Figura 4.13(b) mostra o mapa de localização de adulterações.



(a) Marca d'água extraída.



(b) Mapa de localização de adulterações.

Figura 4.13: Detecção da alteração no tamanho da imagem Lenna.

Compressão da Imagem Marcada

Este teste mostra que a marca d'água é frágil a algoritmos de compressão de imagens. A Figura 4.14 mostra a imagem Bárbara comprimida com fator de qualidade 85%, utilizando JPEG.

Além do erro na extração da marca d'água, o erro na verificação da assinatura digital também indica a presença de adulteração.

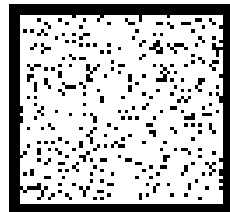
Ataque Vetor de Quantização

Este tipo de ataque tenta forjar uma nova imagem marcada (uma montagem) a partir de imagens marcadas com a mesma marca d'água (com a mesma chave). As porções destas imagens autênticas são combinadas preservando as suas posições relativas.

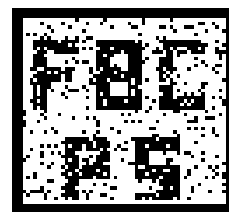
Para construir a imagem falsificada, foram geradas quatro imagens resultantes de pequenas modificações na imagem original. Estas imagens foram marcadas e a nova imagem



(a) Imagem Barbara comprimida.



(b) Marca d'água extraída.



(c) Mapa de localização de adulterações.

Figura 4.14: Fragilidade da marca d'água a compressão.

construída. A Figura 4.15 mostra em (a) a imagem original, em (b) a imagem forjada, em (c) a marca d'água extraída e, em (d) o mapa de localização de adulterações.

Neste caso, as adulterações são detectadas, tanto na marca d'água, extraída com erro, quanto na verificação da assinatura, porém não são localizadas. Mesmo com pequenas modificações, cada imagem modificada possui uma marca d'água única, por causa da assinatura digital. A combinação destas imagens também deve ter uma assinatura digital única, o que garante que este tipo de ataque seja detectado.

Modificação de Pixels na Marca D'água

Para realizar este teste, os *bits* da marca d'água, 010_2 e 110_2 , de alguns pixels da imagem Lenna foram substituídos por 001_2 . O ataque foi detectado na extração da marca d'água e as regiões adulteradas podem ser localizadas por meio do mapa de localização, como mostra a Figura 4.16.

Além da detecção e localização por meio da marca d'água, há falha na verificação da

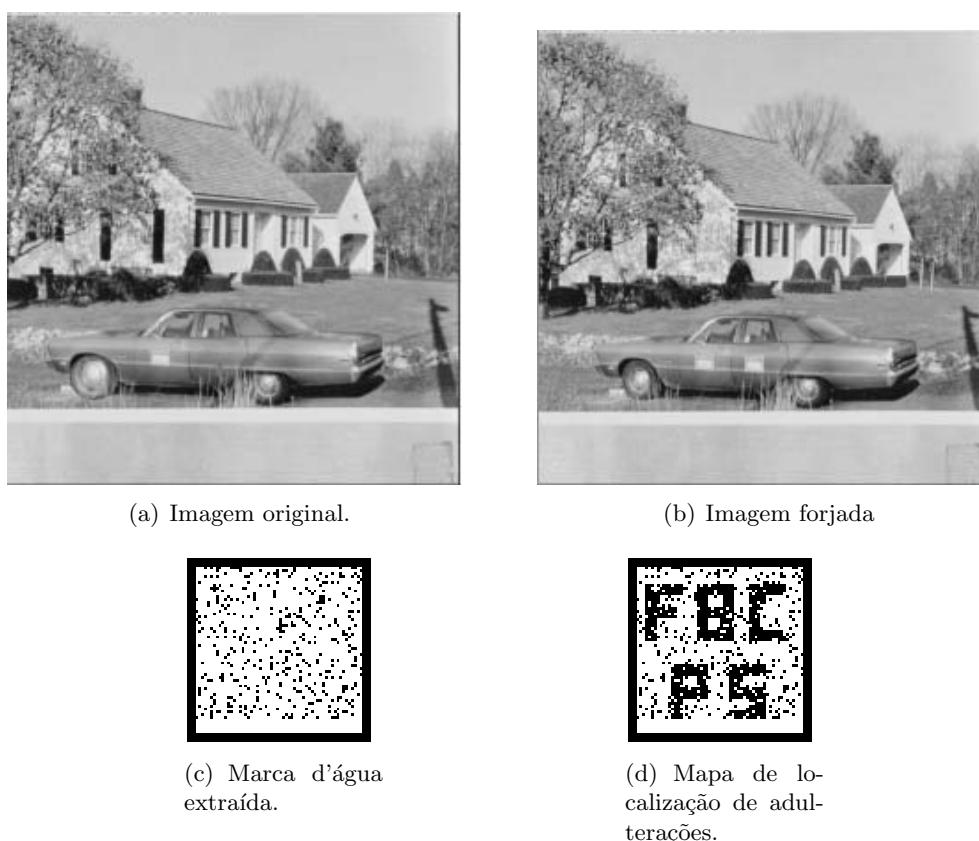


Figura 4.15: Detecção do ataque vetor de quantização

assinatura digital.

Outros Ataques

Outros ataques que devem ser levados em consideração são:

1. Remoção da marca d'água. O método de inserção substitui os *bits* menos significativos da imagem original por novos valores, assim, só é possível remover a marca d'água caso a imagem original seja conhecida. Substituindo os pixels alterados pelos originais. Neste caso, é sempre possível remover a marca d'água, independentemente do método.
2. Alterar a imagem sem prejudicar a marca d'água. Qualquer alteração na imagem resulta em uma alteração no código *hash*. Para alterar a imagem marcada sem prejudicar a marca d'água é necessário que o código *hash* da imagem alterada seja igual ao da imagem marcada. A probabilidade de que exista alguma alteração na imagem, que resulte em um código *hash* válido é muito pequena.

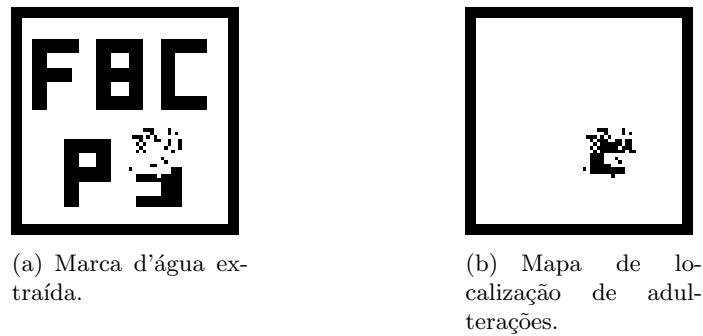


Figura 4.16: Detecção do ataque à marca d'água.

Por causa das propriedades das funções *hash strong collision resistance* e *weak collision resistance*, apresentadas na seção 4.6.2, é computacionalmente impossível encontrar duas entradas diferentes que produzam o mesmo código *hash*, ou encontrar uma determinada entrada que produza um código *hash* igual a um outro existente.

Embora as funções *hash* possuam essas propriedades, existe uma técnica conhecida como o ataque do aniversário (*birthday attacks*) que pode ser utilizada para encontrar uma imagem adulterada que produza um código *hash* igual ao da imagem autêntica. Com um código de 160 *bits*, o esforço requerido para encontrar um par de mensagens (x,y) tal que $H(x) = H(y)$ é da ordem de $2^{160/2} = 2^{80}$ [18]. Além disso, as variações produzidas devem ser tais que a imagem adulterada não seja distorcida e não perca o sentido, o que requer um esforço adicional considerável.

3. Mascaramento. Utilizar a marca d'água de uma imagem autêntica para inserir em outras imagens quaisquer. No método apresentado, o uso da assinatura digital garante que a marca d'água de cada imagem seja única. Assim, ao utilizar o algoritmo de inserção para inserir a marca d'água de uma imagem em outra qualquer, mesmo que estas sejam muito semelhantes, fará com que o processo de verificação da assinatura resulte em falha na autenticação. O atacante não pode gerar uma marca d'água válida, pois não possui a chave privada do transmissor da imagem.
4. Adulterar uma imagem marcada e inserir a marca d'água válida novamente. Nesse caso, a parte significativa da marca d'água é extraída corretamente, pois não tem nenhum vínculo com a imagem. Porém, a adulteração é detectada durante a verificação da assinatura digital, uma vez que esta é vinculada à imagem autêntica.

O falsificador não pode gerar uma nova assinatura digital, pois não possui a chave privada do proprietário da imagem.

Um teste realizado com a imagem Elaine, mostrada na Figura 2(d) no Capítulo Anexo A, mostra que ocorre falha na verificação da assinatura digital. A marca d'água e o mapa de localização não detectam as adulterações, porque a marca d'água é inserida na imagem já modificada, de forma que não é possível detectar e localizar as modificações.

Além destes ataques, também foram avaliadas as seguintes situações:

1. Extrair a marca d'água utilizando outra chave pública. Neste caso, como a chave pública não corresponde à chave privada utilizada na criação da marca d'água, ocorre erro na verificação da assinatura digital.
2. Extrair a marca d'água de uma imagem sem marca d'água. Neste caso, a marca d'água é extraída com erro e é indicado erro de verificação da assinatura. A marca d'água extraída e o mapa de localização de adulterações são semelhantes aos da Figura 4.13.

Estes resultados mostram que o uso apropriado da assinatura digital garante a verificação pública da marca d'água, a verificação de propriedade e garantia de integridade da imagem, pois qualquer modificação na imagem resulta em uma alteração na marca d'água que notifica o receptor.

4.6.3 Propriedades da Proposta de Assinatura Digital para Imagens Desenvolvida

Nesta seção resumem-se as principais propriedades do método de autenticação de imagens por meio de marca d'água apresentado:

- A marca d'água é invisível ao HVS;
- A marca d'água é publicamente verificável, e que resolve diversos problemas relacionados com o compartilhamento de chaves secretas;
- A marca d'água é capaz de localizar adulterações na imagem;

- A verificação da assinatura digital identifica qualquer alteração, bem como o uso da chave pública incorreta, marcas d'água inválidas, e a ausência de marca d'água na imagem;
- A parte significativa da marca d'água localiza adulterações na imagem como cortar e colar, alteração do tamanho, compressão, e mudanças nas propriedades da escala de cinza.

Capítulo 5

Conclusões

As imagens digitais são muito utilizadas em aplicações computacionais e podem ser compartilhadas e distribuídas em sítios na *Internet* com grande facilidade. Essas imagens são utilizadas em uma grande variedade de acervos digitais, sítios de publicação de notícias e de comércio eletrônico, em ambientes distribuídos, em diversas aplicações em rede tais como aplicações médicas etc.

Porém, a rede *Internet* é um canal aberto, de forma que as informações ficam sujeitas a diversos ataques, como interrupção, interceptação, modificação ou fabricação de informações falsificadas por entidades não autorizadas. Além da insegurança dos canais de comunicação, atualmente, existem diversas ferramentas específicas para manipulação de imagens digitais. Muitos aproveitam-se desses dois fatores e conseguem, facilmente, fazer cópias ilegais ou alterar o conteúdo das imagens.

A marca d'água digital é uma solução para coibir a falsificação e distribuição ilegal de imagens digitais. Especificamente, a marca d'água frágil é desenvolvida para garantir a integridade e a autenticidade destas imagens, possibilitando detectar e, em alguns casos, localizar alterações numa imagem previamente marcada. Esta classe de marca d'água foi o alvo da pesquisa e do desenvolvimento desta dissertação.

O objetivo principal da proposta de assinatura digital para imagens desenvolvida é oferecer, por meio do uso da marca d'água, segurança àqueles que utilizam a *Internet* como meio de disponibilização e tráfego de imagens digitais. Ao elaborar esta proposta, cinco fatores foram considerados: a segurança da marca d'água, que designa a sua eficácia contra ataques, a sua compatibilidade com o HVS, a sua complexidade computacional, a

possibilidade de sua verificação pública, e a sua capacidade de localização das adulterações.

A marca d'água desenvolvida é constituída de uma imagem significativa de 64x52 pixels, como um logotipo, e de uma imagem não significativa de 64x8 pixels. Esta imagem não significativa é construída a partir dos *bits* da assinatura digital da imagem, que, por sua vez, é gerada em função dos coeficientes da imagem.

A parte significativa da marca d'água tem como funcionalidade a detecção e a localização de adulterações. Já a parte não significativa garante uma marca d'água segura e única para cada imagem. Por meio da assinatura digital, a marca d'água, garante a integridade, a autenticidade e o não-repúdio da imagem.

Outro aspecto importante é que a própria natureza da assinatura digital garante a verificação pública da marca d'água, resolvendo os diversos problemas de segurança que envolvem o compartilhamento de chaves secretas, utilizadas na maioria dos esquemas de marca d'água publicados. Assim, qualquer pessoa pode verificar a integridade e autenticidade de imagens marcadas sem a necessidade de compartilhar uma chave secreta.

Para garantir sua invisibilidade, a marca d'água é inserida nos coeficientes *wavelet* de maior energia, que correspondem às regiões da imagem menos sensíveis à adição de sinal, o que reduz significativamente as distorções visíveis ao olho humano em consequência da inserção da marca d'água. Além disso, o número de *bits* substituídos na inserção da marca d'água foi reduzido de cinco para três, o que também contribuiu para aumentar a sua invisibilidade. Tal fato é comprovado mediante a comparação dos resultados apresentados nesta dissertação com os do método original apresentados em [28].

Os resultados obtidos comprovam a invisibilidade da marca d'água. Não ocorreram distorções visualmente perceptíveis, o que pode ser conferido por meio dos altos valores de PSNR. Além disso, comparações com outros valores de PSNR obtidos com alguns métodos *blind* que utilizam a DWT mostram que a qualidade visual da imagem marcada é superior com o método apresentado nesta dissertação, conforme mostra a Seção 4.6.1.

A complexidade computacional do método é consideravelmente reduzida em razão do uso da DWT ao invés da DCT ou da DFT. Além disso, o uso do esquema *lifting* contribui substancialmente para a redução do tempo e do espaço em memória gastos na implementação dos passos da transformada *wavelet* [15]. Com essas duas ferramentas matemáticas, o resultado é uma implementação com menor complexidade computacional.

Os resultados apresentados no Capítulo 4 mostram que, em geral, o tempo gasto, tanto com a inserção quanto com a extração da marca d'água, é menor que um minuto.

A execução de testes que simulam vários ataques à marca d'água mostram que todos os ataques foram detectados por meio da verificação da assinatura digital. Com relação à localização, os ataques como compressão ou alteração no tamanho da imagem e o ataque vetor de quantização geram ruído na marca d'água extraída; os ataques que modificam casualmente a imagem marcada e os que danificam a marca d'água, por meio da alteração de seus *bits*, são bem localizados no mapa de detecção de adulterações; já os ataques como utilizar a marca d'água de uma imagem autêntica para inseri-la em quaisquer outras imagens e adulterar uma imagem marcada e inserir a marca d'água válida novamente são detectadas por meio da verificação da assinatura digital, porém as regiões adulteradas não são localizadas.

Por causa da baixa complexidade computacional, esse método é adequado para uso em servidores com grande número de requisições. Para melhorar a segurança, pode ser utilizado conjuntamente com um sistema de autenticação mútua entre o receptor e o transmissor.

Os coeficientes da imagem são inteiros, em consequência disso, o método é apropriado para aplicações que requerem processamento de sinal sem perda de informações e é compatível com a maioria das aplicações para processamento digital de imagens [8]. Além disso, o uso de coeficientes inteiros facilita a implementação do método em *hardware*, podendo ser embarcado em dispositivos ou equipamentos eletrônicos tais como câmeras fotográficas digitais e telefones móveis.

Assim, a aplicação do método não fica restrita à segurança na *Internet*, estendendo sua aplicabilidade para diversas circunstâncias em que é necessário comprovar a autenticidade e integridade de imagens privadas, por exemplo, para checar imagens a serem publicadas em noticiários ou de imagens a serem apresentadas com propósitos legais.

Para trabalhos futuros, sugere-se a alteração dos algoritmos de construção e de inserção da marca d'água para que ela possibilite localizar adulterações feitas na imagem quando a marca d'água válida é inserida em uma imagem qualquer ou em uma imagem marcada adulterada.

Outra sugestão é um estudo do uso das técnicas aplicadas na proposta apresentada em

áudio e vídeo. Assim pode-se estender esta idéia, aplicando as técnicas em certificação digital para vídeo e TVs que usam transmissão de sinais digitais, além, é claro, de dispositivos móveis de comunicação.

O método proposto garante a autenticidade somente de imagens em tons de cinza, pois aplica os algoritmos de construção, inserção e extração da marca d'água apenas aos componentes de luminância da imagem. Caso a imagem seja colorida, apenas os seus componentes de luminância são autenticados. Dessa forma, pode-se alterar os componentes de crominância, sem que a adulteração seja detectada na extração da marca d'água.

No entanto, os mesmos algoritmos podem ser facilmente aplicados aos componentes de crominância da imagem, para que qualquer adulteração em cada componente seja detectada e localizada apropriadamente. Essa medida garante a integridade e a autenticidade dos componentes individualmente, mas não detecta a permutação destes componentes.

Uma possível solução para esse problema, que pode ser facilmente implementada, é calcular a assinatura digital em função dos coeficientes destes componentes em conjunto, ao invés de individualmente. Essa é uma sugestão para melhorar a proposta apresentada nesta dissertação.

Referências Bibliográficas

- [1] KENT S. ; ATKINSON, R. *Security Architecture for the Internet Protocol*. United States: RFC Editor, 1998.
- [2] ATKINS D.; STALLINGS, W. Z. P. *PGP Message Exchange Formats*. United States: RFC Editor, 1996.
- [3] KIM, H. Y. Marcas d'água frágeis de autenticação para imagens em tonalidade contínua e esteganografia para imagens binárias e meio-tom. *RITA*, v. 10, n. 1, p. 97–125, 2003.
- [4] XIAOYUN W.; JUNQUAN, H. Z. G. J. H. A secure semi-fragile watermarking for image authentication based on integer wavelet transform with parameters. In: *ACSW Frontiers '05: Proceedings of the 2005 Australasian workshop on Grid computing and e-research*. Darlinghurst, Australia: Australian Computer Society, Inc., 2005. p. 75–80. ISBN 1-920-68226-0.
- [5] BARNI M.; BARTOLINI, F. P. A. Improved wavelet-based watermarking through pixel-wise masking. *IEEE Transactions on Image Processing*, v. 10, n. 5, p. 783–791, 2001. Disponível em: <<http://dx.doi.org/10.1109/83.918570>>. Acesso em: 03 set. 2007.
- [6] ADDISON, P. S. *The Illustrated Wavelet Transform Handbook Introductory Theory and Applications in Science, Engineering, Medicine and Finance*. [S.l.]: Institute of Physics Publishing Bristol and Philadelphia, 2002.
- [7] DAUBECHIES I.; SWELDENS, W. *Factoring Wavelet Transforms into Lifting Steps*. [S.l.], 1996. Disponível em: <<http://cm.bell-labs.com/who/wim/papers/factor.ps.gz>>. Acesso em: 12 jan. 2007.

- [8] CALDERBANK R.; DAUBECHIES, I. W. S. B.-L. Y. *Wavelet transforms that map integers to integers*. [S.l.], 1996. Disponível em: <citeseer.ist.psu.edu/407090.html>. Acesso em: 12 jan. 2007.
- [9] GOMES J.; VELHO, L. *From Fourier Analysis to Wavelets*. 1th. ed. Instituto de Matemática Pura e Aplicada, IMPA, Rio de Janeiro, Brazil: Prentice Hall, Inc., 1999.
- [10] MALLAT, S. *A Wavelet Tour of Signal Processing*. [S.l.]: Academic Press, 1998.
- [11] TEOLIS, A. *Computational Signal Processing with Wavelets*. [S.l.]: Springer-Verlag Birkhäuser, 1998. 332 p.
- [12] CIÊNCIA, S. da. *As Séries de Fourier - Tintim por TinTim*. Disponível em: <<http://www.searadaciencia.ufc.br/tintim/matematica/fourier/fourier1.htm>>. Acesso em: 15 jun. 2007.
- [13] SCHEINERMAN, E. R. *Matemática Discreta: Uma Introdução*. 1th. ed. [S.l.]: Thomson Learning, 2003.
- [14] DAWEI Z.; GUANRONG, C. W. L. A chaos-based robust wavelet-domain watermarking algorithm. In: *Chaos, Solitons and Fractals*. Elsevier, 2004. p. 47–54. Disponível em: <www.elsevier.com/locate/chaos>.
- [15] SWELDENS W.; SCHRÖDER, P. Building your own wavelets at home. In “Wavelets in Computer Graphics”, ACM SIGGRAPH Course Notes. 1996. Disponível em: <<http://cm.bell-labs.com/who/wim/papers/athome.ps.gz>>. Acesso em: 23 set. 2007.
- [16] UYTTERHOEVEN G.; ROOSE, D. B. A. *Wavelet transforms using lifting scheme*. 1997. Disponível em: <citeseer.ist.psu.edu/uytterhoeven97wavelet.html>. Acesso em: 12 jan. 2007.
- [17] KUROSE JAMES F.; ROSS, K. W. *Redes de computadores e a Internet: uma abordagem top-down*. 3th. ed. Av. Ermano Marchetti, 1435, São Paulo - SP, Brasil: Pearson Education, Inc., 1999. 143-222, 255-275 p.
- [18] STALLINGS, W. *Cryptography And Network Security Principles and Practice*. 2th. ed. Upper Saddle River, New Jersey 07458: Prentice Hall, Inc., 1999.

- [19] FRIEDMANN, G. L. The trustworthy digital camera: Restoring credibility to the photographic image. *IEEE Transactions on Consumer Electronics*, v. 39, n. 4, p. 905–910, nov. 1993. ISSN 0098-3063. Disponível em: <<http://www.cl.cam.ac.uk/fapp2/steganography/bibliography/043125.html>>.
- [20] VOLOSHYNOVSKIY S.; PEREIRA, S. P. T. E.-J. S. J. Attacks on digital watermarks: Classification, estimation-based attacks, and benchmarks. *Communications Magazine, IEEE*, v. 39, n. 8, p. 118–126, ago. 2001. ISSN 0163-6804.
- [21] LIN E.; DELP, E. A review of fragile image watermarks. In: *Proc. of the Multimedia and Security Workshop (ACM Multimedia '99)*. Orlando: [s.n.], 1999. p. 25–29. Disponível em: <citeseer.ist.psu.edu/article/lin99review.html>. Acesso em: 03 set. 2007.
- [22] MACQ B. M.; QUISQUATER, J.-J. B.-S. K. M. Cryptology for digital tv broadcasting. In: *Proceedings of the IEEE*. [S.l.: s.n.], 1995. v. 83, p. 944–957. ISBN 0018-9219.
- [23] COX I. J.; MILLER, M. L. A review of watermarking and the importance of perceptual modeling. In: *Proc. of Electronic Imaging '97*. [s.n.], 1997. Disponível em: <[ftp://ftp.nj.nec.com/pub/ingemar/papers/ei97.zip](http://ftp.nj.nec.com/pub/ingemar/papers/ei97.zip)>. Acesso em: 03 set. 2007.
- [24] DING K.; HE, C. J. L.-G. W. H.-X. Wavelet-based semi-fragile watermarking with tamper detection. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, Oxford University Press, Oxford, UK, E88-A, n. 3, p. 787–790, 2005. ISSN 0916-8508.
- [25] ZHAO Y.; CAMPISI, P. K. D. Dual domain watermarking for authentication and compression of cultural heritage images. *IEEE Trans. Image Processing*, v. 13, n. 3, p. 430–448, mar. 2004. Disponível em: <<http://dx.doi.org/10.1109/TIP.2003.821552>>. Acesso em: 01 dez. 2006.
- [26] CHUN-SHIEN L.; HONG-YUAN, M. L. C.-J.-S. Combined watermarking for image authentication and protection. In: *IEEE International Conference on Multimedia and Expo (III)*. [s.n.], 2000. p. 1415–1418. Disponível em: <citeseer.ist.psu.edu/lu00combined.html>. Acesso em: 01 dez. 2006.
- [27] FRIDRICH J.; GOLJAN, M. B. A. C. New fragile authentication watermark for images. In: *ICIP*. [S.l.: s.n.], 2000.

- [28] HUANG D.; LIU, J. H. J. L. H. A robust dwt-based blind data hiding algorithm. In: *Circuits and Systems*. IEEE Computer Society, 2002. Disponível em: <<http://ieeexplore.ieee.org/>>. Acesso em: 23 set. 2007.
- [29] HARTUNG F.; BERND, G. Fast public-key watermarking of compressed video. In: *International Conference on Image Processing (ICIP'97)*. Santa Barbara, California, U.S.A.: [s.n.], 1997. I, p. 528–531. ISBN 0-8186-8183-7. Disponível em: <citeseer.ist.psu.edu/hartung97fast.html>. Acesso em: 03 set. 2007.
- [30] BHATTACHARJEE S.; KUTTER, M. Compression tolerant image authentication. In: *International Conference on Image Processing*. [s.n.], 1998. p. 435–439. Disponível em: <citeseer.ist.psu.edu/article/bhattacharjee98compression.html>. Acesso em: 01 dez. 2006.
- [31] WONG, P. A public key watermark for image verification and authentication. In: *International Conference on Image Processing*. [s.n.], 1998. p. 455–459. Disponível em: <citeseer.ist.psu.edu/wong98public.html>. Acesso em: 01 dez. 2006.
- [32] MCLAGAN M.; JORDAN, M. *Linux Online*. Ogdensburg, New York, USA: [s.n.]. Disponível em: <<http://www.linux.org/>>. Acesso em: 12 fev. 2008.
- [33] VOLKERDING P.; CANTRELL, D. L. C. *The Slackware Linux Project*. Disponível em: <<http://www.slackware.com/>>. Acesso em: 12 fev. 2008.
- [34] ORGANIZATION, I. L. K. *The Linux Kernel Archives*. Disponível em: <<http://www.kernel.org/pub/linux/kernel/v2.6/>>. Acesso em: 12 fev. 2008.
- [35] TEAM, G. *GCC, the GNU Compiler Collection*. Disponível em: <<http://gcc.gnu.org/>>. Acesso em: 12 fev. 2008.
- [36] SOURCEFORGE. *QccPack*. Disponível em: <<http://qccpack.sourceforge.net>>. Acesso em: 12 fev. 2008.
- [37] ENGELSCHALL, R. S. *OpenSSL: The Open Source toolkit for SSL/TLS*. Disponível em: <<http://www.openssl.org>>. Acesso em: 12 fev. 2008.

- [38] COELHO FERNANDA B.; BARBAR, J. S. C. G. S. B. d. The use of watermark and hash function for the authentication of digital images mapped through the use of the wavelet transform. *Internet and Web Applications and Services, 2007. ICIW '07. Second International Conference on*, p. 48–48, Mai. 2007.
- [39] VALENS, C. *EZW encoding*. 1999–2004. Disponível em: <<http://perso.orange.fr/polyvalens/clemens/ezw/ezw.html>>. Acesso em: 27 set. 2006.
- [40] MULTIMEDIA. *Multimedia Systems and Applications - Image Quality Computation*. Disponível em: <<http://bmrc.berkeley.edu/courseware/cs294/fall97/assignment/psnr.html>>. Acesso em: 23 nov. 2007.
- [41] WEBMASTER, T. K. *K Desktop Environment*. Disponível em: <<http://www.kde.org/>>. Acesso em: 12 fev. 2008.
- [42] KILE. *Kile - an integrated LaTeX environment*. Disponível em: <<http://kile.sourceforge.net/>>. Acesso em: 12 fev. 2008.
- [43] KATE. *Kate-Get an Edge in Editing*. Disponível em: <<http://kate.kde.org/>>. Acesso em: 15 fev. 2008.
- [44] YU, G. J. Mean quantization blind watermarking for image authentication. In: *International Conference on Image Processing*. [s.n.], 2000. p. Vol III: 706–709. Disponível em: <<http://ieeexplore.ieee.org/>>. Acesso em: 01 dez. 2006.

Apêndice A

Códigos fonte dos aplicativos para inserção e extração da marca d'água

Os códigos fonte apresentados neste apêndice são os dos aplicativos que fazem a inserção e a extração da marca d'água, descritos no Capítulo 4.

```
/******
  InsereWatermark.c - Aplicativo que insere a marca d'água
  Fernanda Barbosa Coelho Mendes
  10 de Janeiro de 2008
  fernanda@facom.ufu.br
  *****/
#include <libQccPack.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <openssl/ripemd.h>
#include <openssl/bio.h>
#include <openssl/err.h>
#include <openssl/rsa.h>
#include <openssl/evp.h>
#include <openssl/x509.h>
#include <openssl/pem.h>
#include <openssl/bn.h>
#include <sys/time.h>
#include <openssl/sha.h>
#include "keys.c"

#define MICRO_PER_SECOND      1000000
#define USG_STRING "[-w %s:wavelet] [-b %s:boundary] [-l %d:last_subband]
%s:imgfile %s:subband_pyramid %s:watermak %s:outputimgfile"

#define WTSIZE 64

/* declaracao das variaveis relacionadas com imagens */
QccIMGImage Image1; //Imagem a ser marcada
QccIMGImage Watermark; //Marca d'água original
QccIMGImage OutputWatermark; //Marca d'água extraída
QccIMGImage OutputImage; //Imagem marcada
/* Arquivo em que a marca d'água é escrita*/
QccString OutputWM_Filename = "/mnt/dados/pesquisa/imagens/OutputMask.pbm";

int ImageNumCols; //numero de colunas da imagem
int ImageNumRows; //numero de linhas da imagem

/* declaracao das variaveis relacionadas com wavelets */
```

```

QccWAVWavelet Wavelet;
QccWAVWavelet WaveletInv;
QccString WaveletFilename;
QccString Boundary = "symmetric";

/* Declaracao das variaveis que contem os nomes do
   arquivo que contém a chave privada */
char *fileKey = "/root/.ssh/privkey.pem";
QccWAVSubbandPyramidInt SubbandPyramid;

int NumLevels = 1;
int NumSubbands;
int LastSubband = -1;
int subband;

/* Declaracao de variaveis de proposito geral */
char *ProgramName;
int i,j,val;

/* sigret armazena a assinatura gerada e siglen armazena o seu tamanho*/
char *sigret;
int siglen;
/* Função que calcula a DWT da imagem Image*/
int dwt(QccIMGImage Image){

    if (NumLevels < 1){
        QccErrorAddMessage("%s: Number of levels must be 1 or greater", ProgramName);
        QccErrorExit();
    }

    /* Calcula o numero de sub-bandas a partir do número de níveis de
       decomposição. NumSubbands=3*NumLevels +1 */
    NumSubbands = QccWAVSubbandPyramidIntNumLevelsToNumSubbands(NumLevels);
    if (QccWAVWaveletCreate(&Wavelet, WaveletFilename, Boundary)){
        QccErrorAddMessage("%s: Error calling QccWAVWaveletCreate()", ProgramName);
        QccErrorExit();
    }

    /* Lê a imagem para uma estrutura QccIMGImage. */
    if (QccIMGImageRead(&Image)){
        QccErrorAddMessage("%s: Error calling QccIMGImageRead(%s)", ProgramName, Image.filename);
        QccErrorExit();
    }

    /* Verifica se a imagem é colorida. */
    if (QccIMGImageColor(&Image)){
        QccErrorAddMessage("%s: Color images not supported", ProgramName);
        QccErrorExit();
    }

    /* Inicializa diversos campos das estruturas QccIMGImage e QccWAVSubbandPyramidInt*/
    ImageNumCols = Image.Y.num_cols;
    ImageNumRows = Image.Y.num_rows;
    SubbandPyramid.num_levels = 0;
    SubbandPyramid.num_rows = ImageNumRows;
    SubbandPyramid.origin_row = 0;
    SubbandPyramid.origin_col = 0;
    SubbandPyramid.num_cols = ImageNumCols;
    Wavelet.lifting_scheme.scheme = QCCWAVLIFTINGScheme_IntCohenDaubechiesFeauveau9_7;

    if (QccWAVSubbandPyramidIntAlloc(&SubbandPyramid)){
        QccErrorAddMessage("%s: Error calling QccWAVSubbandPyramidInitialize()", ProgramName);
        QccErrorExit();
    }

    /* Carrega a imagem para a matriz de coeficientes*/
    for(i=0; i<ImageNumRows; i++)
        for(j=0; j<ImageNumCols; j++)
            SubbandPyramid.matrix[i][j] = (int)floor(Image.Y.image[i][j]);

    /* Aplica a DWT, NumLevels=3 */
    if(QccWAVSubbandPyramidIntDWT(&SubbandPyramid,NumLevels, &Wavelet)){

```

```

        QccErrorAddMessage("%s: Error calling QccWAVSubbandPyramidDWT()", ProgramName);
        QccErrorExit();
    }

    /* Esta função seleciona a região S e calcula o codigo hash desta região*/
    if(geraAssinatura()){
        QccErrorAddMessage("%s: Error calling calculaHash()", ProgramName);
        QccErrorExit();
    }

    /* Esta função insere a marca d'água nos coeficientes contidos em subband */
    if (Embedd_Watermark(&SubbandPyramid, subband)){
        QccErrorAddMessage("%s: Error calling EmbeddWatermark()", ProgramName);
        QccErrorExit();
    }

    /* Escreve a matriz de coeficientes em um arquivo */
    if (QccWAVSubbandPyramidIntWrite(&SubbandPyramid))
    {
        QccErrorAddMessage("%s: Error calling QccWAVSubbandPyramidIntWrite()", ProgramName);
        QccErrorExit();
    }

    /* Libera a variável Image*/
    QccIMGImageFree(&Image);
    return 0;
}

/* Esta função calcula a IDWT*/
int idwt(void){

    /* Variável auxiliar para armazenar a matriz de coeficientes resultantes da DWT*/
    QccWAVSubbandPyramidInt Subband_Pyramid;
    char pausa;

    if (QccWAVSubbandPyramidIntInitialize(&Subband_Pyramid)){
        QccErrorAddMessage("%s: Error calling QccWAVSubbandPyramidInitialize()", ProgramName);
        QccErrorExit();
    }

    /* Faz a inicialização de campos da estrutura Subband_Pyramid */
    QccStringCopy(Subband_Pyramid.filename, SubbandPyramid.filename);
    Subband_Pyramid.num_rows = SubbandPyramid.num_rows;
    Subband_Pyramid.num_cols = SubbandPyramid.num_cols;

    /* Alocação de memória para a estrutura */
    if (QccWAVSubbandPyramidIntAlloc(&Subband_Pyramid)){
        QccErrorAddMessage("%s: Error calling QccWAVSubbandPyramidInitialize()", ProgramName);
        QccErrorExit();
    }

    /* Aloca as estruturas do lifting ou banco de filtros e inicializa os campos dessas estruturas. */
    if (QccWAVWaveletCreate(&WaveletInv, WaveletFilename, Boundary)){
        QccErrorAddMessage("%s: Error calling QccWAVWaveletCreate()", ProgramName);
        QccErrorExit();
    }

    /* Carrega para Subband_Pyramida, a matriz salva na função */
    if (QccWAVSubbandPyramidIntRead(&Subband_Pyramid)){
        QccErrorAddMessage("%s: Error calling QccWAVSubbandPyramidIntRead()", ProgramName);
        QccErrorExit();
    }

    NumSubbands = QccWAVSubbandPyramidIntNumLevelsToNumSubbands(Subband_Pyramid.num_levels);

    printf("\nNumSubbands = %d\n", NumSubbands);

    /* Calcula a IDWT*/
    if (QccWAVSubbandPyramidIntInverseDWT(&Subband_Pyramid, &WaveletInv)){
        QccErrorAddMessage("%s: Error calling QccWAVSubbandPyramidIntInverseDWT()", ProgramName);

```

```

    QccErrorExit();
}

OutputImage.Y.num_rows = ImageNumRows;
OutputImage.Y.num_cols = ImageNumCols;

QccIMGImageAlloc(&OutputImage);
/* Após o cálculo da IDWT a imagem reconstruída está armazenada em Subband_Pyramid,
   devendo ser copiada para uma estrutura QccIMGImage */
for(i=0; i<ImageNumRows; i++)
    for(j=0; j<ImageNumCols; j++)
        OutputImage.Y.image[i][j] = Subband_Pyramid.matrix[i][j];
/* Define as dimensões da imagem*/
if (QccIMGImageSetMaxMin(&OutputImage))
{
    QccErrorAddMessage("%s: Error calling QccIMGImageSetMaxMin()", ProgramName);
    QccErrorExit();
}
/* Escreve a imagem em um arquivo */
if (QccIMGImageWrite(&OutputImage))
{
    QccErrorAddMessage("%s: Error calling QccIMGImageWrite()", ProgramName);
    QccErrorExit();
}

    QccWAVSubbandPyramidIntFree(&Subband_Pyramid);
    return 0;
}

/* Esta função gera a assinatura digital da imagem a ser marcada */
int geraAssinatura(void){

    char *regiaoS;
    char *password = NULL;
    char *engine = NULL;
    int row,col,i,j,k,formatKey;
    int subbandCols, subbandRows;
    char *hash;
    RSA *rsa=NULL;
    EVP_PKEY *pkey;
    BIO *bio_err=NULL;
    ENGINE *e = NULL;
    FILE* fpass;
    char saux[10];
    /* Calcula o tamanho da sub-banda LL3 */
    if(QccWAVSubbandPyramidIntSubbandSize(&SubbandPyramid,0,&subbandRows,&subbandCols)){
        QccErrorAddMessage("%s: Error calling QccWAVSubbandPyramidIntSubbandSize()", ProgramName);
        QccErrorExit();
    }
    /* aloca espaço para regiaoS */
    regiaoS = (char *)malloc(((ImageNumRows * ImageNumCols) - (subbandRows * subbandCols))*sizeof(char)*10);

    if(regiaoS == NULL){
        printf("\nERRO DE ALOCACAO DE MEMORIA\n");
        return 1;
    }

    /* Copia os coeficientes das sub-bandas HL3, LH3, HH3, HL2, LH2, HH2, HL1, LH1, HH1
       para regiaoS, os valores inteiros são convertidos em strings*/
    k=0;
    for (row = 0; row < ImageNumRows; row++){
        for (col = 0; col < ImageNumCols; col++){
            if(!(row < subbandRows && col < subbandCols)){
                sprintf(saux,"%u",SubbandPyramid.matrix[row][col]);
                j=strlen(saux);
                for(i=0; i<j;i++){
                    regiaoS[k]=saux[i];
                    k++;
                }
                regiaoS[k]='\0';
            }
        }
    }
}

```

```

    }
}
/* A função RIPEMD160() calcula o código hash e o armazena em hash */
hash = (char *)malloc((RIPEMD160_DIGEST_LENGTH)*sizeof(char)+1);
hash = RIPEMD160(regiaoS, strlen(regiaoS),hash);
formatKey=FORMAT_PEM;
/* Cria a estrutura que armazenará erros gerados durante a carga da chave */
if (bio_err == NULL)
    if ((bio_err=BIO_new(BIO_s_file())) != NULL)
        BIO_set_fp(bio_err,stderr,BIO_NOCLOSE|BIO_FP_TEXT);

/* Lê a chave privada de um arquivo para a variável pkey*/
password=NULL;
pkey = load_key(bio_err, fileKey,formatKey,0,password,e,"Private Key");

if (pkey != NULL)
    rsa = EVP_PKEY_get1_RSA(pkey);
EVP_PKEY_free(pkey);

if (rsa != NULL){
    sigret = (char *) (malloc(RSA_size(rsa))+1);
    /* Assina o código hash utilizando a chave privada*/
    if((RSA_sign(NID_ripemd160, hash, RIPEMD160_DIGEST_LENGTH, sigret, &siglen, rsa)) == 0){
        printf("\n---ERRO NA GERAÇÃO DA ASSINATURA---\n");
        return 1;
    }
}
else{
    printf("\n---ERRO AO CARREGAR A CHAVE---\n");
    return 1;
}
return 0;
}

/* Esta função insere a marca d'água substituindo os três LSBs*/
int replace_bits_e(int watermark_pixel, int coeficiente_pixel){

int resultW;
int resultI;
int x = watermark_pixel;
int y;

/* resultW=1 se o bit da marca d'água é um, caso contrario é zero. */
resultW = x & 1;
/* Obtem os três LSBs*/
resultI = coeficiente_pixel & 0x07;

/* Inserção da marca d'água */
if(resultW){
    if(resultI <= 2){
        y = coeficiente_pixel - 2;
        coeficiente_pixel = ((y & ~0x07) | 6);
    }
    else{
        y = coeficiente_pixel;
        coeficiente_pixel = ((y & ~0x07) | 6);
    }
}
else{
    if(resultI >= 6){
        y = coeficiente_pixel + 4;
        coeficiente_pixel = ((y & ~0x07) | 2);
    }
    else{
        y = coeficiente_pixel;
        coeficiente_pixel = ((y & ~0x07) | 2);
    }
}

/* retorna o coeficiente com a marca d'água inserida*/
return coeficiente_pixel;

```

```
}

/* Função que insere a marca d'água na imagem */
int Embedd_Watermark(QccWAVSubbandPyramidInt *subband_pyramid, int Subband){

    int row, col, num_rows, num_cols, wrow,wcol;
    int num_subbands;
    int subband_num_rows;
    int subband_num_cols;
    int subband_row;
    int subband_col;
    long long int watermark_pixel;
    int i=0;
    char *assinatura;
    char assinatura_char;

    if (subband_pyramid == NULL)
        return(0);
    if (subband_pyramid->matrix == NULL)
        return(0);
    if (subband < 0)
        return(0);

    assinatura = (char *)malloc(128);
    assinatura = sigret;
    num_subbands = QccWAVSubbandPyramidIntNumLevelsToNumSubbands(subband_pyramid->num_levels);

    /*
    Testa se a sub-banda especificada não excede o número de sub-bandas que a imagem foi dividida. */
    if (Subband > num_subbands){
        QccErrorAddMessage("Specified subband is large for given subband pyramid");
        return(1);
    }

    /* Le o arquivo que contem a marca d'água */
    if(QccIMGImageRead(&Watermark)){
        QccErrorAddMessage("%s: Error calling QccIMGImageRead()", ProgramName);
        QccErrorExit();
    }

    if (QccIMGImageColor(&Watermark))
    {
        QccErrorAddMessage("%s: Color images not supported", ProgramName);
        QccErrorExit();
    }

    /* Verifica o tamanho da sub-banda Subband na piramide subband_pyramid. O tamanho da sub-banda
    é retornado em subband_num_rows e subband_num_cols. Retorna 1 caso haja erro.*/
    if (QccWAVSubbandPyramidIntSubbandSize(subband_pyramid,
                                           Subband,
                                           &subband_num_rows,
                                           &subband_num_cols)){
        QccErrorAddMessage("Error calling QccWAVSubbandPyramidIntSubbandSize()");
        return(1);
    }

    /* Calcula a localização da sub-banda Subband em subband_pyramid->matrix,
    os offsets de linha e de coluna, com relação ao elemento superior esquerdo da matriz,
    são retornados em subband_row e em subband_col, respectivamente. */
    if (QccWAVSubbandPyramidIntSubbandOffsets(subband_pyramid,
                                              Subband,
                                              &subband_row,
                                              &subband_col)){
        QccErrorAddMessage("Error calling QccWAVSubbandPyramidIntSubbandOffsets()");
        return(1);
    }

    /*Este trecho insere os bits da imagem significativa da marca d'água*/
    for (row = 0; row < WTSIZE - 8; row++){
        for (col = 0; col < WTSIZE; col++){
```

```

        watermark_pixel = (long long int)Watermark.Y.image[row][col];
        subband_pyramid->matrix[row][col] = replace_bits_e(watermark_pixel,subband_pyramid->matrix[row][col]);
    }
}

/*Este trecho insere os bits da assinatura digital*/
i=0;
for (row = WTSIZE - 8;row < WTSIZE; row++){
    for (col = 0; col < WTSIZE; col++){
        assinatura_char = assinatura[i];
        i++;
        for(j=0; j<8; j++){
            if(assinatura_char & 0x01){
                subband_pyramid->matrix[row][col] = replace_bits_e(1,subband_pyramid->matrix[row][col]);
            }
            else{
                subband_pyramid->matrix[row][col] = replace_bits_e(0,subband_pyramid->matrix[row][col]);
            }
            assinatura_char=assinatura_char>>1;
            if(j < 7){
                if(col < WTSIZE)
                    col++;
                else{
                    col=0;
                    row++;
                }
            }
        }
    }
}
return(0);
}

/* Função principal */
int main(int argc, char *argv[]){

    int i;
    float delta;
    /* Declaração das variaveis utilizadas no calculo do tempo
       gasto na inserção da marca d'água*/
    struct timeval inicio;
    struct timeval fim;

    QccInit(argc, argv);
    /* Inicializa as estruturas*/
    QccWAVWaveletInitialize(&Wavelet);
    QccWAVWaveletInitialize(&WaveletInv);
    QccIMGImageInitialize(&Image1);
    QccIMGImageInitialize(&Watermark);
    QccIMGImageInitialize(&OutputImage);
    QccIMGImageInitialize(&OutputWatermark);
    if (QccWAVSubbandPyramidIntInitialize(&SubbandPyramid)){
        QccErrorAddMessage("%s: Error calling QccWAVSubbandPyramidInitialize()", ProgramName);
        QccErrorExit();
    }

    /* Quebra a string passada como argumento, copiando cada uma para a respectiva variável*/
    if (QccParseParameters(argc, argv,
        USQ_STRING,
        WaveletFilename,
        Boundary,
        &LastSubband,
        Image1.filename,
        SubbandPyramid.filename,
        Watermark.filename,
        OutputImage.filename))

        QccErrorExit();

    QccStringCopy(OutputWatermark.filename,OutputWM_Filename);

```

```

QccStringCopy(WaveletFilename, "/usr/local/share/QccPack/Wavelets/CohenDaubechiesFeauveau.9-7.int.lft");
ProgramName = (char *) malloc(sizeof(argv[0]));

strcpy(ProgramName, argv[0]);
/* Obtem o tempo inicial*/
gettimeofday(&inicio, NULL);
NumLevels=3;
/* Chama a função para calcular a DWT*/
dwt(Image1);
/* Chama a função para calcular a IDWT*/
idwt();
/* Obtem o tempo final*/
gettimeofday(&fim, NULL);

/* Calcula o tempo (em segundos) gasto na inserção*/
delta = fim.tv_sec - inicio.tv_sec;
delta += ((fim.tv_usec - inicio.tv_usec)/(float)MICRO_PER_SECOND);
printf("\nTempo gasto na insercao: %f seg.\n", delta);

free(ProgramName);
/* Libera a memória alocada para as estruturas*/
QccWAVWaveletFree(&Wavelet);
QccWAVWaveletFree(&WaveletInv);
QccIMGImageFree(&OutputImage);
QccIMGImageFree(&Watermark);
QccIMGImageFree(&OutputWatermark);
QccWAVSubbandPyramidIntFree(&SubbandPyramid);

/* Finaliza a execução*/
QccExit;
}

/*****
ExtraiWatermark.c - Aplicativo que extrai a marca d'água
Fernanda Barbosa Coelho Mendes
10 de Janeiro de 2008
fernanda@facom.ufu.br
*****/
#include <libQccPack.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <openssl/ripemd.h>
#include <openssl/bio.h>
#include <openssl/err.h>
#include <openssl/rsa.h>
#include <openssl/evp.h>
#include <openssl/x509.h>
#include <openssl/pem.h>
#include <openssl/bn.h>
#include <sys/time.h>
#include <openssl/sha.h>
#include "keys.c"

#define MICRO_PER_SECOND 1000000
#define USG_STRING "[-w %s:wavelet] [-b %s:boundary] [-l %d:last_subband] %s:imgfile\n%s:subband_pyramid %s:watermak %s:detection"

#define WTSIZE 64

/* declaracao das variaveis relacionadas ao tratamento de imagens */
QccIMGImage Image1;
QccIMGImage OutputWatermark;
QccIMGImage OutputImage;
QccIMGImage Watermark;
QccIMGImage DetectionImage;
QccString OutputWM_Filename = "/mnt/dados/pesquisa/imagens/OutputMask.pbm";

```

```
int ImageNumCols;//numero de colunas da imagem
int ImageNumRows; //numero de linhas da imagem

/* declaracao das variaveis relacionadas ao tratamento de wavelets */
QccWAVWavelet Wavelet;
QccWAVWavelet WaveletInv;
QccString WaveletFilename;
QccString Boundary = "symmetric";

/* Declaracao das variaveis que contem o nome do arquivo
   que contém a chave pública */
char *fileKey = "/root/.ssh/pubkey.pem";

QccWAVSubbandPyramidInt SubbandPyramid;

int NumLevels = 1;
int NumSubbands;
int LastSubband = -1;
int subband;

/* Declaracao de variaveis de proposito geral */
char *ProgramName;
int i,j,val;
int NUM = 100;

/* assinatura armazena a assinatura gerada e siglen armazena o seu tamanho*/
int siglen = 64;
char *assinatura;

int dwt(QccIMGImage Image){

    char pausa;
    QccString ImageTemp_FileName = "/mnt/dados/pesquisa/imagens/lennaSubbands.pgm";
    /* O número de níveis deve ser maior que um */
    if (NumLevels < 1){
        QccErrorAddMessage("%s: Number of levels must be 1 or greater", ProgramName);
        QccErrorExit();
    }

    /* Calcula o numero de sub-bandas a partir do número de níveis de
       decomposição. NumSubbands=3*NumLevels +1 */
    NumSubbands = QccWAVSubbandPyramidIntNumLevelsToNumSubbands(NumLevels);
    if (QccWAVWaveletCreate(&Wavelet, WaveletFilename, Boundary)){
        QccErrorAddMessage("%s: Error calling QccWAVWaveletCreate()", ProgramName);
        QccErrorExit();
    }
    /* Lê a imagem para uma estrutura QccIMGImage. */
    if(QccIMGImageRead(&Image))
    {
        QccErrorAddMessage("%s: Error calling QccIMGImageRead(%s)", ProgramName, Image.filename);
        QccErrorExit();
    }
    /* Verifica se a imagem é colorida. */
    if (QccIMGImageColor(&Image))
    {
        QccErrorAddMessage("%s: Color images not supported", ProgramName);
        QccErrorExit();
    }
    /* Inicializa diversos campos das estruturas QccIMGImage e QccWAVSubbandPyramidInt*/
    ImageNumCols = Image.Y.num_cols;
    ImageNumRows = Image.Y.num_rows;
    SubbandPyramid.num_levels = 0;
    SubbandPyramid.num_rows = ImageNumRows;
    SubbandPyramid.num_cols = ImageNumCols;
    Wavelet.lifting_scheme.scheme = QCCWAVLIFTINGSCHEME_IntCohenDaubechiesFeauveau9_7;

    if (QccWAVSubbandPyramidIntAlloc(&SubbandPyramid)){
        QccErrorAddMessage("%s: Error calling QccWAVSubbandPyramidInitialize()", ProgramName);
```

```

    QccErrorExit();
}

/* Carrega a imagem para a matriz de coeficientes*/
for(i=0; i<ImageNumRows; i++)
    for(j=0; j<ImageNumCols; j++)
        SubbandPyramid.matrix[i][j] = (int)floor(Image.Y.image[i][j]);

/* Aplica a DWT, NumLevels=3 */
if(QccWAVSubbandPyramidIntDWT(&SubbandPyramid, NumLevels, &Wavelet)){
    QccErrorAddMessage("%s: Error calling QccWAVSubbandPyramidDWT()", ProgramName);
    QccErrorExit();
}

/* Esta função extrai a marca d'água nos coeficientes contidos em subband */
if (Extract_Watermark(&SubbandPyramid, subband)){
    QccErrorAddMessage("%s: Error calling EmbeddWatermark()", ProgramName);
    QccErrorExit();
}

/* Escreve a matriz de coeficientes em um arquivo */
if (QccWAVSubbandPyramidIntWrite(&SubbandPyramid))
{
    QccErrorAddMessage("%s: Error calling QccWAVSubbandPyramidIntWrite()", ProgramName);
    QccErrorExit();
}

QccIMGImageFree(&Image);

return 0;
}

/* Esta funcao calcula o hash da regionS da imagem e
e faz a verificação da assinatura extraída da imagem*/
int verificaAssinatura(void){

    char *regiaoS;
    char *password = NULL;
    char *engine = NULL;
    int row,col,i,j,k,formatKey;
    int subbandCols, subbandRows;
    char *hash;
    RSA *rsa=NULL;
    EVP_PKEY *pkey;
    BIO *bio_err=NULL;
    ENGINE *e = NULL;
    FILE* fpass;
    char aux;
    char saux[10];

    /* Calcula o tamanho da sub-banda LL3 */
    if(QccWAVSubbandPyramidIntSubbandSize(&SubbandPyramid,0,&subbandRows,&subbandCols)){
        QccErrorAddMessage("%s: Error calling QccWAVSubbandPyramidIntSubbandSize()", ProgramName);
        QccErrorExit();
    }

    /* aloca espaço para regiaoS */
    regiaoS = (char *)malloc(((ImageNumRows * ImageNumCols) - (subbandRows * subbandCols))*sizeof(char)*10);

    if(regiaoS == NULL){
        printf("\nERRO DE ALOCAÇÃO DE MEMÓRIA\n");
        return 1;
    }

    k=0;
    for (row = 0; row < ImageNumRows; row++){
        for (col = 0; col < ImageNumCols; col++){
            if(!(row < subbandRows && col < subbandCols)){
                sprintf(saux,"%u",SubbandPyramid.matrix[row][col]);

```

```

        j=strlen(saux);
        for(i=0; i<j;i++){
            regioaS[k]=saux[i];
            k++;
        }
        regioaS[k]='\0';
    }
}

/* A variavel hash armazena o codigo hash calculado a partir de regioaS */
/* A função RIPEMD160() calcula o código hash e o armazena em hash */
hash = (char *)malloc((RIPEMD160_DIGEST_LENGTH)*sizeof(char) +1);
hash = RIPEMD160(regioaS,strlen(regioaS),hash);

formatKey=FORMAT_PEM;
/* Cria a estrutura que armazenará erros gerados durante a carga da chave */
if (bio_err == NULL)
    if ((bio_err=BIO_new(BIO_s_file())) != NULL)
        BIO_set_fp(bio_err,stderr,BIO_NOCLOSE|BIO_FP_TEXT);

password=NULL;
/* Lê a chave pública de um arquivo para a variável pkey*/
pkey = load_pubkey(bio_err, fileKey,formatKey,0,password,e,"Public Key");

if (pkey != NULL)
    rsa = EVP_PKEY_get1_RSA(pkey);
else
    return 1;

EVP_PKEY_free(pkey);
/* Faz a verificação da assinatura utilizando a chave pública */
if (rsa != NULL){
    if((RSA_verify(NID_ripemd160,hash,RIPEMD160_DIGEST_LENGTH,assinatura,siglen,rsa)) == 0){
        printf("\n\n---ERRO NA VERIFICAÇÃO DA ASSINATURA---\n\n");
        return 2;
    }
}
else{
    printf("\n\n---ERRO AO CARREGAR A CHAVE---\n\n");
    return 1;
}

return 0;
}

/* Extrai os bits que farão parte da imagem significativa
da marca d'água extraída */
int replace_bits_x(int coeficiente_pixel){

int result0;
int result1;
int resultI;
int watermark_pixel;

/* Obtem os três LSBs*/
resultI = coeficiente_pixel & 0x07;
result1 = resultI ^ 6; /*Se LSB = 6 (representação do bit 0) */
result0 = resultI ^ 2; /*Se LSB = 2 (representação do bit 1) */

if(!result0)
    watermark_pixel= 0;
else
    watermark_pixel= 1;

return watermark_pixel;
}

```

```

/* Extrai os bits que farão parte da imagem não significativa
da marca d'água extraída, que constitui a assinatura digital */
char replace_bits_x1(int coeficiente_pixel){

int result0;
int result1;
int resultI;
char watermark_pixel;
int y = coeficiente_pixel;

resultI = y & 0x07;
result1 = resultI ^ 6; /*Se LFB (5 bits menos significativos) = 24 (representação do bit 1) */
result0 = resultI ^ 2; /*Se LFB (5 bits menos significativos) = 8 (representação do bit 1) */

    if(!result0)
        watermark_pixel= '0';

    else
        watermark_pixel= '1';

    return watermark_pixel;

}

/* Extrai a marca d'água da imagem */
int Extract_Watermark(QccWAVSubbandPyramidInt *subband_pyramid, int Subband){

    int row, col;
    int num_subbands;
    int subband_num_rows;
    int subband_num_cols;
    int subband_row;
    int subband_col;
    long long int watermark_pixel;
    int i=0;
    char assinatura_char;
    char aux;
    char bitVec[8] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
    char temp[64];

    if (subband_pyramid == NULL)
        return(0);
    if (subband_pyramid->matrix == NULL)
        return(0);
    if (subband < 0)
        return(0);

    num_subbands = QccWAVSubbandPyramidIntNumLevelsToNumSubbands(subband_pyramid->num_levels);

    /*
    Testa se a sub-banda especificada não excede o número de sub-bandas que a imagem foi dividida.
    */
    if (Subband > num_subbands){
        QccErrorAddMessage("Specified subband is large for given subband pyramid");
        return(1);
    }

    /* Verifica o tamanho da sub-banda Subband na piramide subband_pyramid.
    O tamanho da sub-banda é retornado em subband_num_rows e subband_num_cols.
    Retorna 1 caso haja erro. */
    if (QccWAVSubbandPyramidIntSubbandSize(subband_pyramid,
                                           Subband,
                                           &subband_num_rows,
                                           &subband_num_cols)){
        QccErrorAddMessage("Error calling QccWAVSubbandPyramidIntSubbandSize()");
        return(1);
    }

    /* Calcula a localização da sub-banda Subband em subband_pyramid->matrix,
    os offsets de linha e de coluna, com relação ao elemento superior esquerdo da matriz,

```

```

    são retornados em subband_row e em subband_col, respectivamente.  */
if (QccWAVSubbandPyramidIntSubbandOffsets(subband_pyramid,
                                           Subband,
                                           &subband_row,
                                           &subband_col)){
    QccErrorAddMessage("Error calling QccWAVSubbandPyramidIntSubbandOffsets()");
    return(1);
}

/* Inicializa campos da imagem que receberá a marca d'água extraída*/
OutputWatermark.Y.num_rows = WTSIZE;
OutputWatermark.Y.num_cols = WTSIZE;
QccIMGImageSetSize(&OutputWatermark,WTSIZE,WTSIZE);
QccIMGImageAlloc(&OutputWatermark);

/* Extrai os bits da imagem significativa */
for (row = 0; row < WTSIZE - 8; row++){
    for (col = 0; col < WTSIZE; col++){
        OutputWatermark.Y.image[row][col] = replace_bits_x(subband_pyramid->matrix[row][col]);
    }
}

for (row = WTSIZE - 8; row < WTSIZE; row++){
    for (col = 0; col < WTSIZE; col++){
        OutputWatermark.Y.image[row][col] = 1;
    }
}

/* aloca memória para a assinatura digital*/
assinatura = (char *)malloc(128);
i=0;
assinatura_char = 0x00; //inicialização

/* monta a assinatura digital a partir dos bits de marca d'água extraídos*/
for (row = WTSIZE - 8; row < WTSIZE; row++){
    for (col = 0; col < WTSIZE; col++){
        /* Calcula um caractere */
        for(j=0; j<8; j++){
            aux = replace_bits_x1(subband_pyramid->matrix[row][col]);
            if(aux & 0x01)
                assinatura_char = assinatura_char ^ bitVec[j];
            if(j < 7){
                if(col < WTSIZE)
                    col++;
                else{
                    col=0;
                    row++;
                }
            }
        }
        /* Depois de calculado, o caractere é armazenado na assinatura digital*/
        assinatura[i] = assinatura_char;
        assinatura_char = 0x00;
        i++;
    }
}

i = verificaAssinatura();
if(i == 1){

    QccErrorAddMessage("%s: Error calling verificaAssinatura()", ProgramName);
    QccErrorExit();
}
else
    if(i == 2)
        printf("-----> A IMAGEM FOI ADULTERADA <-----\n\n");

/* Define as dimensões da imagem significativa */
if (QccIMGImageSetMaxMin(&OutputWatermark)){
    QccErrorAddMessage("%s: Error calling QccIMGImageSetMaxMin()", ProgramName);
    QccErrorExit();
}

```

```

}

/* Escreve imagem significativa (parte da marca d'água em um arquivo)*/
if (QccIMGImageWrite(&OutputWatermark)){
    QccErrorAddMessage("%s: Error calling QccIMGImageWrite()", ProgramName);
    QccErrorExit();
}

/* Esta função calcula o mapa de localização de adulterações*/
if(geraDtetectionImage()){
    QccErrorAddMessage("%s: Error calling QccIMGImageWrite()", ProgramName);
    QccErrorExit();
}

return(0);
}

/* Esta função calcula o mapa de localização de adulterações*/
int geraDtetectionImage(void){

    int row,col;
    int tamper = 0;
    /* Le o arquivo que contem a marca d'água */
    if(QccIMGImageRead(&Watermark)){
        QccErrorAddMessage("%s: Error calling QccIMGImageRead()", ProgramName);
        QccErrorExit();
    }

    if (QccIMGImageColor(&Watermark))
    {
        QccErrorAddMessage("%s: Color images not supported", ProgramName);
        QccErrorExit();
    }

    DetectionImage.Y.num_rows = WTSIZE;
    DetectionImage.Y.num_cols = WTSIZE;
    QccIMGImageSetSize(&DetectionImage,WTSIZE,WTSIZE);
    QccIMGImageAlloc(&DetectionImage);
    DetectionImage.image_type = QCCIMGTYPE_PBM;

    /* Localiza os pixels adulterados, para construir o mapa de localização */
    for (row = 0;row < WTSIZE - 8; row++){
        for (col = 0; col < WTSIZE; col++){
            if(Watermark.Y.image[row][col] != OutputWatermark.Y.image[row][col]){
                DetectionImage.Y.image[row][col] = 0;
                tamper++;
            }
            else
                DetectionImage.Y.image[row][col] = 1;
        }
    }

    for (row = WTSIZE - 8;row < WTSIZE; row++)
        for (col = 0; col < WTSIZE; col++)
            DetectionImage.Y.image[row][col] = 1;

    if(tamper > 0)
        printf("---- NÚMERO DE PIXELS ADULTERADOS: %d\n", tamper);

    if (QccIMGImageSetMaxMin(&DetectionImage)){
        QccErrorAddMessage("%s: Error calling QccIMGImageSetMaxMin()", ProgramName);
        QccErrorExit();
    }

    if (QccIMGImageWrite(&DetectionImage)){
        QccErrorAddMessage("%s: Error calling QccIMGImageWrite()", ProgramName);
        QccErrorExit();
    }
}

```

```

    return 0;
}

/* Função principal */
int main(int argc, char *argv[]){

    int i;
    char pausa;
    float delta;
    /* Declaração das variaveis utilizadas no calculo do tempo
       gasto na extração da marca d'água*/
    struct timeval inicio;
    struct timeval fim;

    QccInit(argc, argv);
    /* Inicializa as estruturas*/
    QccWAVWaveletInitialize(&Wavelet);
    QccWAVWaveletInitialize(&WaveletInv);
    QccIMGImageInitialize(&Image1);
    QccIMGImageInitialize(&OutputImage);
    QccIMGImageInitialize(&DetectionImage);
    QccIMGImageInitialize(&Watermark);
    QccIMGImageInitialize(&OutputWatermark);
    if (QccWAVSubbandPyramidIntInitialize(&SubbandPyramid)){
        QccErrorAddMessage("%s: Error calling QccWAVSubbandPyramidInitialize()", ProgramName);
        QccErrorExit();
    }

    /* Quebra a string passada como argumento, copiando cada uma para a respectiva variável*/
    if (QccParseParameters(argc, argv,
        USG_STRING,
        WaveletFilename,
        Boundary,
        &LastSubband,
        Image1.filename,
        SubbandPyramid.filename,
        Watermark.filename,
        DetectionImage.filename))
        QccErrorExit();

    QccStringCopy(OutputWatermark.filename, OutputWM_Filename);
    QccStringCopy(WaveletFilename, "/usr/local/share/QccPack/Wavelets/CohenDaubechiesFeauveau.9-7.int.lft");
    ProgramName = (char *) malloc(sizeof(argv[0]));
    strcpy(ProgramName, argv[0]);

    /* Obtem o tempo inicial*/
    gettimeofday(&inicio, NULL);

    NumLevels=3;
    dwt(Image1);
    /* Obtem o tempo final*/
    gettimeofday(&fim, NULL);
    /* Calcula o tempo (em segundos) gasto na extração*/
    delta = fim.tv_sec - inicio.tv_sec;
    delta += (fim.tv_usec - inicio.tv_usec)/(float)MICRO_PER_SECOND;
    printf("\nTempo gasto na extracao: %.6f seg.\n", delta);

    /* Libera a memória alocada para as estruturas*/
    free(ProgramName);
    QccWAVWaveletFree(&Wavelet);
    QccWAVWaveletFree(&WaveletInv);
    QccIMGImageFree(&OutputImage);
    QccIMGImageFree(&DetectionImage);
    QccIMGImageFree(&Watermark);
    QccIMGImageFree(&OutputWatermark);

    QccWAVSubbandPyramidIntFree(&SubbandPyramid);
    /* Finaliza a execução*/
    QccExit; }

```

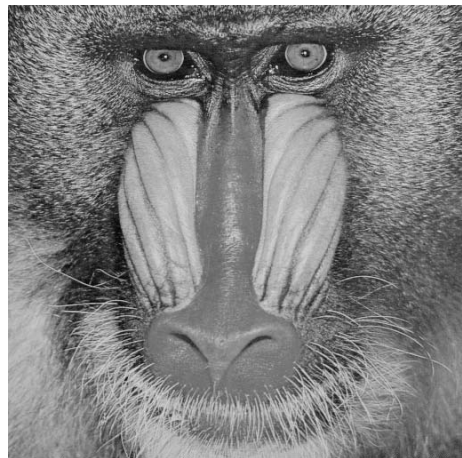
Anexo A

Figuras utilizadas para testes

O Capítulo 4 apresenta diversos testes realizados com o aplicativo que implementa o método proposto nesta dissertação. Os testes foram feitos com imagens normalmente utilizadas por pesquisadores para avaliar métodos de compressão de imagens e de inserção de marca d'água. Este anexo mostra as figuras que foram utilizadas nestes testes.



(a) Avião



(b) Baboon

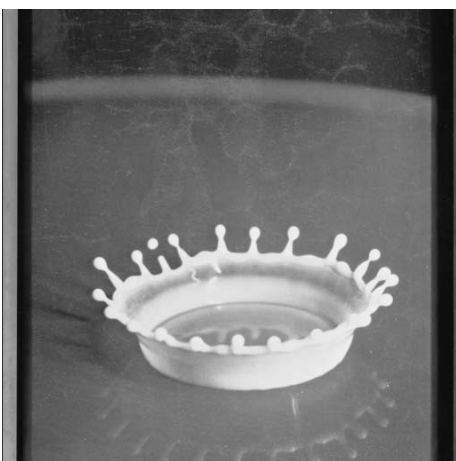
Figura 1: Figuras utilizadas nos testes.



(a) Badlands



(b) Barbara



(c) Coroa



(d) Elaine

Figura 2: Figuras utilizadas nos testes (continuação).



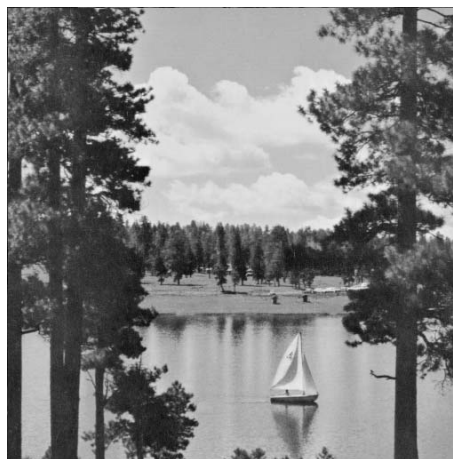
(a) F16



(b) Goldhill



(c) House



(d) Lago

Figura 3: Figuras utilizadas nos testes (continuação).



(a) Lenna



(b) Peppers



(c) Ponte



(d) Tanque1

Figura 4: Figuras utilizadas nos testes (continuação).



(a) Tanque2



(b) Tanque3



(c) Tanque4



(d) Tanques

Figura 5: Figuras utilizadas nos testes (continuação).



(a) Tiffany



(b) Truck

Figura 6: Figuras utilizadas nos testes (continuação).