ESCOLA POLITÉCNICA - UNIVERSIDADE DE SÃO PAULO
Departamento de Engenharia de Sistemas Eletrônicos
Avenida Professor Luciano Gualberto, travessa 3, nº158 CEP: 05508-900 São Paulo SP
Telefone: (0xx-11) 3091.5728 Fax (0xx-11) 3091.5585

RELATÓRIO PSI-2594

INICIAL ( )
INTERMEDIÁRIO ( )
FINAL (X)

DATA DE ENTREGA: ___/___/ 2007

NOME DO ALUNO: DANILO ZUCOLLI FIGUEIREDO

Nº USP: 4943009

LOCAL DO PROJETO: São Paulo - SP

NOME DO ORIENTADOR: PROF. DR. HAE YONG KIM

TÍTULO DO PLANO DE TRABALHO: MASCARAMENTO DE INFORMAÇÕES E AUTENTICAÇÃO DE IMAGENS COLORIDAS POR MARCAS D'ÁGUA REVERSÍVEIS

OBJETIVO DO PLANO DE TRABALHO: TEXTO QUE DETALHA O PROJETO DESENVOLVIDO

SUMÁRIO

RESUMO....2
1 INTRODUÇÃO....3
2 OBJETIVOS DO PROJETO....8
3 METODOLOGIA....10
4 MODELAGEM DA SOLUÇÃO....14
5 RESULTADOS....30
6 DISCUSSÃO....36
7 CONCLUSÕES....38
BIBLIOGRAFIA....39

LISTA DE ILUSTRAÇÕES

FIGURA 1 – PRIMEIRO NÍVEL DA WBS DO PROJETO 10
FIGURA 2 – WBS DA FASE DE INICIAÇÃO DO PROJETO 10
FIGURA 3 – WBS DA FASE DE PLANEJAMENTO DO PROJETO 11
FIGURA 4 – WBS DA FASE DE EXECUÇÃO DO PROJETO 12
FIGURA 5 – WBS DA FASE DE ENCERRAMENTO DO PROJETO 12
FIGURA 6 – VIZINHANÇA DO PIXEL EM PROCESSAMENTO 16
FIGURA 7 – REPRESENTAÇÃO ESQUEMÁTICA DA ELABORAÇÃO DA IMAGEM MARCADA 20
FIGURA 8 - REPRESENTAÇÃO ESQUEMÁTICA DA VERIFICAÇÃO DA AUTENTICIDADE DA IMAGEM 21
FIGURA 9 – DIAGRAMA DE CASOS DE USOS 22
FIGURA 10 – DIAGRAMA DE CLASSES SIMPLIFICADO 27
FIGURA 11 - INTERFACE GRÁFICA 1 28
FIGURA 12 - INTERFACE GRÁFICA 2 28
FIGURA 13 - BOB MARLEY.BMP 31
FIGURA 14 - BONECOS.BMP 31
FIGURA 15 - CAT.BMP 32
FIGURA 16 - EASTER ISLAND.BMP 32
FIGURA 17 - FAMÍLIA.BMP 32
FIGURA 18 - GOURD DRUM.BMP 32
FIGURA 19 - JARDIM.BMP 32
FIGURA 20 - LENA.BMP 32
FIGURA 21 - MONALISA.BMP 33
FIGURA 22 - PAPAGAIOS.BMP 33
FIGURA 23 - PIPOCA.BMP 33
FIGURA 24 - VAN GOGH.BMP 33

LISTA DE TABELAS

TABELA 1 – CLASSES IMPLEMENTADAS.....26
TABELA 2 – TEMPO DE PROCESSAMENTO E PSNR.....34
TABELA 3 – ANÁLISE DE PSNR EM FUNÇÃO DO NÍVEL DE QUANTIZAÇÃO.....35

RESUMO

Este projeto descreve uma implementação de técnica de mascaramento de informações e autenticação de imagens coloridas por marcas d'água reversíveis para o caso de imagens coloridas não-compactadas, tendo sido elaborado um protótipo de *software* que implementa a função de autenticação proposta fazendo uso de um algoritmo de criptografia assimétrico.

A solução foi baseada nos algoritmos SHA 256, RSA e na marca d'água digital *Lossless Generalized-LSB Data Embedding*, que juntos formam um sistema de autenticação de imagens digitais.

São descritas a modelagem e a metodologia do projeto, bem como as fases da implementação, os resultados e é feita uma discussão acerca dos materiais e métodos utilizados.

1 INTRODUÇÃO

Processamento digital de imagens

O Processamento Digital de Imagens tem como focos principais a melhoria de informações visuais para interpretação humana e o processamento e análise de imagens para percepção de máquinas.

Diversas são as áreas de aplicação de processamento de imagens. Alguns exemplos são: astronomia, biologia, geologia, geografia, medicina, direito e inspeção industrial.

Na área da indústria de produção de mídia, questões como propriedade, integridade e autenticação de dados digitais têm recebido grande atenção, principalmente devido ao maciço compartilhamento de arquivos via internet. Uma técnica de Processamento Digital de Imagens vem sendo amplamente aplicada nesse contexto: marcas d'água digitais.

Esteganografia e Marcas d'água digitais

Esteganografia e Marcas d'água digitais são conceitos relacionados a ocultamento de informações, mas que apresentam aplicações bastante diferentes.

Esteganografia é a área de estudo preocupada com as técnicas que podem ser usadas para embutir, de maneira pouco visível, informações em uma dada mensagem ou meio.

Técnicas de Esteganografia baseiam-se fundamentalmente na invisibilidade à detecção, tendo por objetivo esconder uma mensagem oculta dentro de uma informação hospedeira. Observadores ao analisarem esta informação não seriam capazes de detectar a mensagem secreta. Apenas observadores que conhecessem o método de decodificação seriam capazes de enxergar o que foi ocultado dentro da informação hospedeira.

Já os métodos que envolvem Marcas d'água (*Watermarking*) visam associar à informação hospedeira uma assinatura ou dado relevante. Conforme KIM e AFIF (2003), marca d'água digital é um sinal de interesse embutido de modo visualmente pouco perceptível em uma imagem digital. A atribuição desse sinal permite a

obtenção *a posteriori* de informações adicionais relevantes sobre a imagem hospedeira ou sobre algum domínio que se relacione a essa imagem.

Aplicações de marcas d'água não se limitam a sinais do tipo imagens, podendo ser usadas também em sons e vídeos, por exemplo.

O grau de capacidade de uma marca d'água se manter válida após a imagem marcada ter sido submetida a algum tipo de processamento, ou seja, a dificuldade de se remover uma marca, permite classificá-la como marca robusta, semifrágil ou frágil.

As robustas são aquelas que resistem à maioria dos procedimentos de manipulação de imagens enquanto as marcas frágeis são facilmente corrompidas. Já as semifrágeis são marcas d'água que resistem, por definição, a apenas transformações que não modificam o conteúdo visual da imagem.

Aplicações como *copyright* e verificação de propriedade demandam marcas robustas capazes de suportar manipulações, já aplicações como autenticação e verificação de integridade demandam marcas frágeis, pois qualquer alteração na imagem deve ser detectada.

A autenticação de imagens garante que aquela analisada seja idêntica a seu original e pode igualmente assegurar a identidade de sua fonte.

Uma das técnicas mais difundidas em Marcas d'água envolve a alteração do *bit* menos significativo (LSB – *Least Significant Bit*) de alguns dos *pixels* da imagem a ser marcada. Em imagens coloridas, essa técnica de alteração do bit menos significativo pode ser realizada em três bits de cada *pixel*, sendo cada um deles representante de uma das componentes do padrão de cores RGB. Outras idéias utilizadas em técnicas de Marcas d'água são *Spread Spectrum*, QIM, *Additive*/*Multiplicative*.

As técnicas de Marcas d´agua para imagens digitais geralmente se valem de alguma técnica de compressão de dados de modo a possibilitar que uma maior quantidade de dados ocultos possam ser embutidos no processo.

Alguns interessantes exemplos de técnicas de marcas reversíveis podem ser encontrados, por exemplo, em ALATTAR (2004), CHANG; TAI e LIN (2005), FRIDRICH et al. (2004), GOLJAN; FRIDRICH e DU (2001), HONSINGER et al. (2001), KALKER e WILLEMS (2003), LEEST; VEEN e BRUEKERS (2003), SHI; ANSARI e WEI (2003), THODI e RODRIGUEZ (2004), TIAN (2002), TSAI et al.

(2004), VLEESCHOUWER; DELAIGLE e MACQ (2003), XUAN et al. (2004), YANG et al. (2004), ZAIN; BALDWIN e CLARKE (2004) e em ZHANG e WANG (2004).

Assinaturas digitais

Em muitas aplicações de Marcas d'água em imagens digitais, a informação embutida, marca, é uma assinatura digital associada à imagem e à fonte de origem da imagem,

Assinatura digital é, por definição, um mecanismo que permite, de forma única e exclusiva, a comprovação da autoria ou consentimento de uma fonte em relação a determinado grupo de dados.

Sua implementação geralmente é feita através de dois elementos principais: um resumo, *message digest*, do dado que será assinado e a cifragem desse resumo por meio de um algoritmo criptográfico.

A aplicação de marcas d'água que embutam assinaturas digitais visa trazer maior segurança à imagem, criando uma proteção contra a manipulação, seja ela intencional ou não, e permitindo também a garantia da autenticidade da imagem.

Criptografia de chave pública

Implementações de assinaturas digitais geralmente fazem uso de algoritmos de criptografia assimétrica para garantir sua segurança. Esses algoritmos fazem uso de uma chave pública que pode ser divulgada, e outra secreta, conhecida somente por um conjunto limitado de pessoas. A chave privada é usada no processo de cifragem da mensagem e a pública no de decifragem.

O primeiro criptossistema prático de chave pública para cifragem e aplicado em assinaturas digitais foi o Rivest, Shamir, Adleman (RSA). Hoje é o mais conhecido e aplicado método criptográfico desse tipo. Sua segurança computacional depende fundamentalmente da dificuldade do problema da fatoração de números inteiros grandes. Outros algoritmos de criptografia assimétrica muito difundidos são Diffie-Hellman, Digital Signature Algorithm, ElGamal e Fortezza.

A criptografia por chave pública viabiliza a comunicação segura entre quaisquer usuários e acaba com o problema da distribuição de chaves existente na

criptografia por chave secreta, pois não há necessidade do compartilhamento de uma mesma chave, nem de um pré-acordo entre as partes envolvidas.

Proposta de projeto e justificativas

Questões como propriedade, integridade e autenticação de dados digitais têm recebido grande atenção, principalmente devido ao maciço compartilhamento de arquivos via internet.

Desse contexto e das Marcas d'água digitais aplicadas à autenticação de imagens surgiu a motivação para o desenvolvimento do presente trabalho. Neste projeto foi desenvolvida uma implementação de técnica de mascaramento de informações e autenticação de imagens coloridas por marcas d'água reversíveis para o caso de imagens coloridas não-compactadas, tendo sido elaborado um protótipo de *software* que implementa a função de autenticação proposta fazendo uso de um algoritmo de criptografia assimétrico.

A técnica de Marca d'água escolhida para o projeto foi a *Lossless Generalized-LSB Data Embedding*, descrita em CELIK et al. (2005). Trata-se de uma generalização da idéia de alteração do bit menos significativo (LSB) dos *pixels* da imagem marcada, de modo que a alteração é realizada em um número maior que um de bits e não apenas no menos significativo.

Em CELIK et al. (2005), é descrito todo o mecanismo para que a marca d'água seja embutida. Nele são envolvidos além do processamento dos valores dos *pixels* das imagens, a compactação dos dados por meio de uma versão adaptada do algoritmo CALIC, WU X. e MEMON N. (1997), e um algoritmo de codificação aritmética de dados.

O projeto desenvolvido apresenta inúmeras aplicações práticas. Notadamente as áreas em que há maior demanda por uma solução desse tipo são as áreas médica, legal e militar.

O projeto apresenta caráter inovador do ponto de vista da aplicação de marcas d'água reversíveis para imagens coloridas, complementando a implementação da *Lossless Generalized-LSB Data Embedding* para imagens em tons de cinza.

Com característica multidisciplinar, o projeto emprega conhecimentos e técnicas das áreas de processamento digital de sinais, teoria da informação, comunicações e criptografia.

2 OBJETIVOS DO PROJETO

A proposta deste projeto é a elaboração de um sistema de processamento de imagens capaz de realizar o mascaramento de informações e autenticação de imagens coloridas por marcas d'água reversíveis. Assim, o protótipo de *software* obtido é uma ferramenta para garantia e verificação de autenticidade de imagens digitais, utilizando-se para isso de técnica de marca d'água reversível e frágil, criptografia assimétrica e compressão de dados.

Um sistema que realiza as operações de processamento de imagens executa usualmente as funções de aquisição, armazenamento, processamento, comunicação e exibição de imagens. Nossa atenção foi concentrada na etapa de processamento e síntese de imagens, no qual marcas d'água digitais melhor se enquadram.

Considerando ainda que o foco do projeto foi o desenvolvimento e a aplicação de uma técnica específica de marca d'água e não o desempenho do sistema obtido, as funções de processamento foram implementadas via *software*, eliminando-se a necessidade do desenvolvimento de um *hardware* especializado.

O sistema desenvolvido é capaz de realizar a inserção e extração da marca d'água, a verificação da autenticidade da imagem marcada e a cifragem e decifragem da mensagem transmitida na marca. Ou seja, trata imagens originais gerando imagens marcadas e trata imagens marcadas verificando sua autenticidade e reconstituindo a imagem original da qual foi criada.

Para tanto, a imagem que se deseja marcar é submetida a uma técnica de esteganografia para imagens coloridas, inserindo na figura tanto as informações para a garantia de sua autenticidade quanto para sua reconstituição ao final do processo. Essas exigências do projeto determinaram a escolha da técnica de marca d'água como sendo de uma marca reversível, capaz de suprir as especificações citadas.

Devido ao intenso desenvolvimento de pesquisas na área de marcas d'água, o escopo desse projeto se limita à implementação de uma marca bem definida e não engloba o desenvolvimento de uma nova técnica. A técnica escolhida para o projeto foi a *Lossless generalized-LSB data embedding*, descrita em CELIK et al. (2005).

A marca proposta em CELIK et al. (2005) é um exemplo de Estado da Arte em termos de marcas reversíveis e atende a todas as especificações do projeto.

Esses foram os fatores preponderantes para sua adoção em detrimento de outras possíveis técnicas.

Considerações de segurança foram feitas durante todo o desenvolvimento, mas dada a velocidade em que surgem novos ataques e procedimentos sob os quais a marca d'água poderá ser submetida, não se pode garantir o período no qual a solução será tida como segura.

Algoritmos de autenticação encontram-se sob constante ameaça de ataques. Estas podem se constituir apenas de simulações ou ataques realmente maliciosos. No primeiro caso, o interesse é verificar o grau de segurança apresentado pelo algoritmo e, no segundo, ocorre uma real ameaça contra o sistema.

Como meta técnica para o projeto foi estabelecido um limite máximo de degradação para as imagens marcadas. Esse limite garante às imagens marcadas uma qualidade mínima, preservando o conteúdo visual da imagem original e tornando a marca quase imperceptível.

Como meta técnica para o projeto foi estabelecido que as imagens marcadas devem apresentar no mínimo 30dB de *Peak Signal-to-Noise Ratio* (PSNR), Razão Sinal/Ruído de Pico.

PSNR é um índice que avalia a diferença entre a imagem antes e depois de seu processamento, sendo que imagens marcadas que preservem em muito as características visuais da imagem original implicam em elevado valor de PSNR.

O Peak Signal-to-Noise Ratio (PSNR) é definido por meio da expressão $PSNR = 10 \cdot \log \frac{J \cdot K \cdot [\max\{F(j,k)\}]^2}{\sum_j \sum_k [F(j,k) - Fm(j,k)]^2}$, onde *F* é a imagem original e *Fm* a imagem marcada.

3 METODOLOGIA

As atividades necessárias para o desenvolvimento do projeto podem ser divididas em termos de fases temporais ou em termos de tipos de tarefas, tendo sido adotada a primeira estratégia. A divisão em fases gera as etapas: iniciação, planejamento, execução e encerramento do projeto.

Podemos descrever os esforços necessários ao projeto de maneira esquemática na representação conhecida como *Work Breakdown Structure* (WBS), como segue:

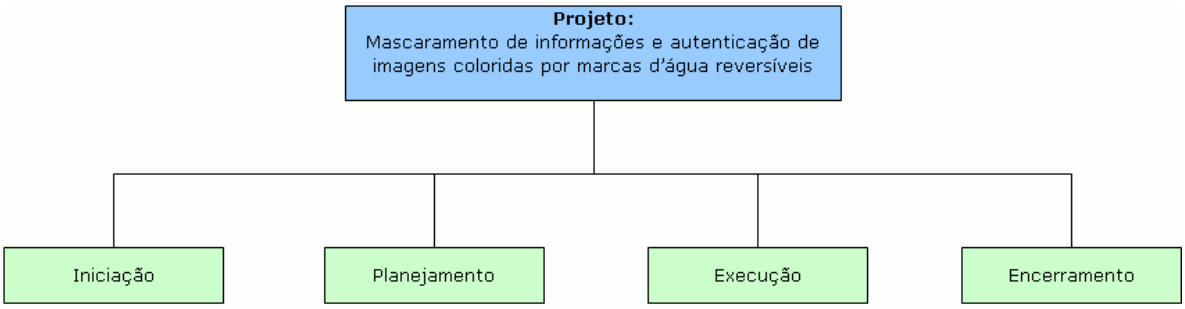

Figura 1 – Primeiro nível da WBS do projeto

A iniciação do projeto constituiu-se na fase marcada por sua concepção e pela definição inicial do escopo do projeto. Aliado a isso houve a etapa de formação do time do projeto e busca pelo Professor Orientador mais adequado ao escopo definido.

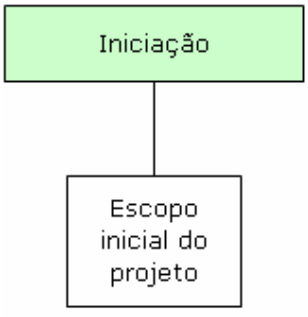

Figura 2 – WBS da fase de iniciação do projeto

Vencida a etapa inicial, passou-se ao planejamento do projeto através de um maior detalhamento do escopo, a definição dos objetivos, metas e metodologia do projeto, além da definição de um cronograma preliminar.

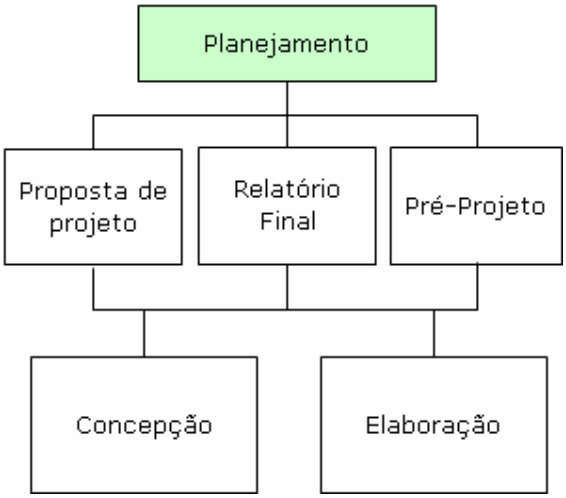

Figura 3 – WBS da fase de planejamento do projeto

Em seguida, na etapa de execução, iniciou-se o detalhamento da solução técnica por meio da definição dos algoritmos a serem implementados e da documentação do projeto por meio de diagramas UML.

Na fase de desenvolvimento a codificação do *software* foi realizada. O desenvolvimento se deu por incrementos e o controle das parcelas desenvolvidas ocorreu de maneira contínua e constante. Para tanto foram realizados processos de controle, ou seja, pausas para que os resultados parciais obtidos fossem testados e o projeto como um todo fosse avaliado.

Dessa maneira, a fase de execução foi marcada fortemente pelas atividades de codificação, documentação e controle do projeto.

Diversos testes foram realizados visando garantir que cada um dos algoritmos tivesse o funcionamento esperado. Também foram testados em conjunto formando um sistema único, de modo a garantir que não houvesse nenhum problema de saída de métodos incompatíveis com entradas de métodos seguintes.

Além disso, diferentes imagens digitais foram testadas. As figuras eram sempre coloridas, mas podiam ser divididas entre tons contínuos e tons discretos. Imagens em tom contínuo são aquelas em que os pixels da imagem apresentam valores de componentes RGB semelhantes aos valores dos pixels de sua

vizinhança, como acontece em fotos ou quadros. Já nas imagens em tons discretos o oposto ocorre e isso pode ser observado em imagens artificiais geradas em *softwares* para criação de imagens, por exemplo.

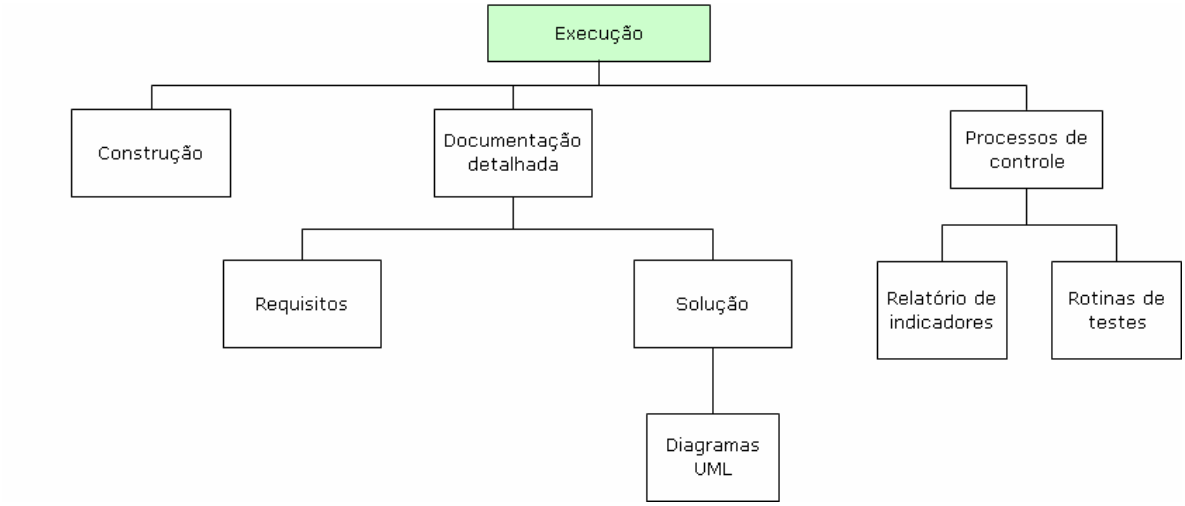

Figura 4 – WBS da fase de execução do projeto

Por último, na fase de encerramento foi feita a validação do *software* através de uma série de testes em diferentes imagens e a entrega do protótipo do projeto foi realizada.

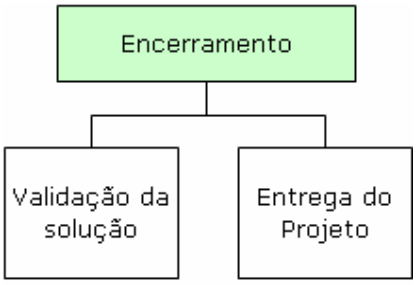

Figura 5 – WBS da fase de encerramento do projeto

O gerenciamento do projeto visou controlar três das principais restrições do projeto - e que usualmente são as restrições de projetos de Engenharia - escopo, tempo e qualidade. A meta durante todo o desenvolvimento foi submeter o projeto ao menor número possível de mudanças de escopo, controlar eventuais atrasos durante as etapas intermediárias e garantir que a entrega final fosse um protótipo que satisfizesse todas as especificações do projeto.

As metodologias PMI, UML e uma série de rotinas de testes para garantia da qualidade forneceram subsídios ao gerenciamento, desenvolvimento, documentação técnica do projeto e controle da qualidade do desenvolvimento.

A linguagem de programação adotada foi a linguagem orientada a objetos C#. Sua escolha foi feita de acordo com o grau de adequação aos requisitos e especificações do projeto e com a disponibilidade de bibliotecas de interesse disponíveis para uso.

C# é uma linguagem orientada a objetos desenvolvida pela Microsoft. Sua sintática é baseada em C++, sendo procedural e orientada a objetos. Ela possui grande influência de outras linguagens como o Delphi e o Java, tendo grande ênfase em tornar a codificação mais simples. C# tornou-se um padrão para as entidades *Ecma International* e *International Organization for Standardization* (ISO).

O projeto foi implementado no ambiente de desenvolvimento Microsoft Visual C# 2005 Express Edition. Trata-se de um ambiente que prove a desenvolvedores as ferramentas e o suporte necessário à codificação na linguagem C#. Permite a criação de aplicações robustas usando toda a infra-estrutura do .NET Framework.

O Microsoft .NET Framework é um componente de software que pode ser adicionado ao sistema operacional Microsoft Windows. Ele provê um extenso conjunto de soluções previamente codificadas que cumprem os requisitos comuns para o desenvolvimento e controla a execução de programas escritos especificamente para esse framework. O .NET Framework pode ser visto como um conjunto de bibliotecas voltadas a diversos tipos de aplicações.

4 MODELAGEM DA SOLUÇÃO

Lossless Generalized-LSB Data Embedding

As abordagens utilizadas no projeto de sistemas de mascaramento de informações e autenticação de imagens podem ser divididas basicamente em três: *Regions of Non-Interest* (RONI), marcas não reversíveis e marcas reversíveis.

O emprego de marcas aplicadas em regiões que não apresentam relevância para a aplicação da imagem (RONI) é caracterizado pela possibilidade de se embutir dados sem que haja a preocupação da não degradação da imagem, visto que a marca em nada afeta a utilização prática da imagem.

Para que se faça uso dessa idéia é necessário deter *a priori* uma função capaz de segmentar a figura dividindo-a em áreas de interesse e em áreas sem importância. A capacidade de se embutir dados (*bits per pixel*) é limitada pela área de não-interesse que, na maioria das aplicações, varia de acordo com a imagem, podendo impedir o uso da técnica em alguns casos.

As marcas não reversíveis quando utilizadas para o mascaramento de informações e autenticação de imagens requerem a especificação de um limite máximo de degradação ao qual a figura pode ser submetida. Neste caso, a capacidade de se embutir dados é proporcional ao limite de degradação imposto. Como a imagem original não pode ser obtida a partir da marcada, existem diversas áreas de aplicação que não podem se valer dessa técnica.

Já as marcas reversíveis apresentam como principal característica a capacidade de se obter a imagem original a partir da marcada. Isso, junto a um parâmetro de qualidade para a imagem marcada, implica na limitação da capacidade de se embutir informações na imagem.

Apesar dessas limitações, as marcas reversíveis são as melhores opções para a implementação de um sistema de mascaramento e autenticação de imagens digitais, pois são mais simples e versáteis que as marcas do tipo RONI e apresentam, diferentemente das não reversíveis, a possibilidade de remoção da marca.

FENG J. et al (2006) classifica as implementações de marcas reversíveis em compressão de dados, *difference expansion* e *histogram bin exchanging*. Elas

diferem pelo modo através do qual as informações da marca e os dados para sua remoção são inseridos na imagem. Empregam-se as técnicas de compressão de dados, expansão da diferença entre o valor de *pixels* vizinhos e deslocamento do histograma de distribuição de valores de *pixels* e exemplos podem ser encontrados em FRIDRICH J., GOLJAN M. e DU R. (2002), TIAN J. (2003) e VLEESCHOUWER C. D., DELAIGLE J. E. e MACQ B. (2001), respectivamente.

Dadas as características propostas e desejadas para o projeto, foi escolhida a técnica de marcas d'água *Lossless Generalized-LSB Data Embedding* proposta em CELIK et al.'s (2005), por ser reversível, frágil, pouco distorcer a imagem hospedeira e possuir capacidade suficiente para embutir os dados necessários para a autenticação.

Essa técnica se vale da idéia de embutir nos *L bits* menos significativos dos valores dos *pixels* da imagens os dados necessários para reconstrução da imagem original a partir da imagem marcada e dados genéricos que se deseja embutir.

Os dados necessários à reconstrução nada mais são que os próprios valores dos *L bits* menos significativos dos valores dos *pixels* da imagem compactados por uma versão adaptada do algoritmo de compressão de imagens digitais sem perdas CALIC.

Por terem sido compactados, nos *L bits* menos significativos dos valores dos *pixels* da imagem sobra espaço para que informações sejam embutidas. Como o presente projeto tem como aplicação a autenticação das imagens, o dado embutido na imagem é sua assinatura digital, ou seja, seu resumo, *message digest*, cifrado por meio de um algoritmo criptográfico assimétrico.

O algoritmo descrito em CELIK et al.'s (2005) foi elaborado para imagens em tons de cinza. Como o projeto é destinado a imagens podemos aplicar o algoritmo para cada um dos três componentes do espaço de cores RGB.

Por simplificação, em seguida faremos a exposição de como é aplicado o algoritmo a um desses componentes, mas deve-se ter em mente que o processo pode ser aplicado aos três, com a vantagem de que com mais componentes pode-se embutir mais dados na imagem.

O processo de inserção ou retirada da marca é iniciado com a quantização da matriz de intensidade dos *pixels*. Essa é uma matriz que apresenta as mesmas dimensões da imagem em processamento e tem em suas diferentes posições os valores das intensidades dos *pixels* da imagem.

A quantização é feita *pixel* a *pixel* através das expressões:

$$s(x,y) = q(x,y) + r(x,y) \quad \text{(I)}$$

$$q(x,y) = L \cdot chão(\frac{s(x,y)}{L}) \quad \text{(II)}$$

Onde *s(x,y)* é o valor da intensidade da cor do *pixel*, *q(x,y)* o valor quantizado, *r(x,y)* o valor do resto, *L* o nível de quantização e o par ordenado *(x,y)* a posição na matriz.

Os passos seguintes do processo visam criar dois contextos, *d*, *θ*, que serão usados na fase de codificação aritmética. Os contextos são descrições da vizinhança em que o *pixel* se insere e a eles é associada uma função de distribuição de probabilidades para os valores de resto *r(x,y)*.

Para a elaboração dos contextos, primeiramente temos o cálculo da predição do *pixel* atual s_0.

$$s_0 = \frac{1}{4} \cdot \sum_{k \in \{W,N,E,S\}} f(s_k) \quad \text{(III)}$$

$$f(s_k) = \begin{cases} s_k \leftarrow k \in \{W, NW, N, NE\} \\ Q_L(s_k) + \frac{L}{2} \leftarrow caso\_contrário \end{cases} \quad \text{(IV)}$$

As referências aos pixels vizinhos àquele que está sendo processado seguem o mapeamento da Fig. 6.

NW	N	NE
W	O	E
SW	S	SE

Figura 6 – Vizinhança do pixel em processamento

Em seguida são calculados os contextos *Δ*, *d* e *t*.

$$\Delta = \sum \frac{1}{8} \cdot \left| f(s_k) - s_0 \right| \quad \text{(V)}$$

$$d = Q(\Delta) \quad \text{(VI)}$$

$$t_k = \begin{cases} 1 \leftarrow f(s_k) > s_0 \\ 0 \leftarrow caso\_contrário \end{cases} \quad \text{(VII)}$$

$$t = t_w \,\|\, t_N \,\|\, t_E \,\|\, t_s \quad \text{(VIII)}$$

A predição é então refinada, utilizando-se para isso o valor do erro médio de predição para o contexto do *pixel* O, $\varepsilon(d,t)$.

$$\bar{s}_0 = round(s_0 + \varepsilon(d,t) \quad \text{(IX)}$$

O último contexto calculado é θ, que é uma função do valor do resto r, do valor quantizado $Q_L(s_0)$ e do nível de quantização L.

$$\theta_0 = \begin{cases} 0 \leftarrow Q_L(s_0) \geq \bar{s}_0 \\ L-1 \leftarrow Q_L(s_0) + L - 1 \leq \bar{s}_0 \\ \bar{s}_0 - Q_L(s_0) \leftarrow caso\_contrário \end{cases} \quad \text{(X)}$$

Para cada contexto (t, θ) é mantida uma função distribuição de probabilidades, pdf(t, θ), para os valores dos restos r. Essa pdf(t, θ) é atualizada conforme os *pixels* da imagem são processados, tanto na etapa da codificação da marca d'água quanto em sua decodificação e extração.

De posse dessas funções é executada a (de)codificação dos valores dos restos através de um algoritmo que implementa a codificação aritmética adaptativa aos contextos seguindo algoritmo descrito em SAYOOD (2000).

Conforme apresentado, a marca descrita em CELIK et al.'s (2005), *Lossless Generalized-LSB Data Embedding,* pode ser descrita em pseudo-linguagem da seguinte maneira:

1) Determinação de $q(x,y)$ e $r(x,y)$

2) s_0 = Predição do pixel atual( )

3) d, t = Determinação do Contexto D, $T(s_0)$

4) s_0' = Predição refinada (s_0, d, t)

5) θ = Determinação do Contexto θ (s_0')

6)

Se ($\theta \geq 0$),

Encode/Decode Resíduo (r, d, θ);

Caso contrário,

Encode/Decode Resíduo ($L - 1 - r$, d, θ);

SHA-256

Segundo DAHAB e LOPEZ (2007), uma função de *Hash* ou função resumo é uma função que calcula uma representação condensada de uma mensagem ou arquivo. Mais precisamente, uma função resumo recebe como entrada uma cadeia de bits de comprimento arbitrário e devolve outra cadeia de bits de comprimento fixo, chamado resumo.

Uma função de resumo criptográfico é uma função H: $\{0,1\}^* \rightarrow \{0,1\}^n$ satisfazendo as seguintes propriedades:

a) Resistência à primeira inversão: Dado um resumo r, é inviável encontrar uma mensagem m tal que r = H(m).
b) Resistência à segunda inversão: Dado um resumo r e uma mensagem m_1 tal que r = H(m_1), é inviável encontrar uma mensagem $m_2 = m_1$ tal que r = H(m_2).
c) Resistência a colisões: Dado um resumo r, é inviável encontrar mensagens m_1, m2 tais que H(m_1) = H(m_2).

A partir dessas propriedades torna-se evidente a aplicação de funções de Hash para a confecção de códigos de detecção de modificações de mensagens, códigos de autenticação e assinaturas digitais.

A função de *Hash* adotada foi o *Secure Hash Algorithm* 256 (SHA-256). Este é um algoritmo já consolidado, amplamente utilizado em aplicações e seguro.

A implementação desse algoritmo encontra-se na biblioteca System.Security.Cryptography do .NET framework. Essa implementação permite que, a partir de um vetor de *bytes* de comprimento qualquer, obtenhamos um vetor de *bytes* de comprimento 32, ou seja, um resumo de 256 *bits*.

O objeto da biblioteca System.Security.Cryptography usado foi o SHA256Managed e os métodos foram Initialize(.) e ComputeHash(.).

RSA

Segundo DAHAB e LOPEZ (2007), o primeiro criptossistema prático de chave pública para encriptação e assinaturas digitais foi o Rivest, Shamir, Adleman (RSA). Sua segurança computacional depende essencialmente da dificuldade do problema da fatoração de números inteiros grandes. Na prática, esse problema é considerado intratável para números de comprimento maior ou igual a 2048 *bits*. A descrição do RSA é baseada nas operações aritméticas modulares em Z_n e na exponenciação modular.

A implementação do RSA também se encontra na biblioteca System.Security.Cryptography do .NET framework. Essa implementação permite que, a partir de um vetor de *bytes*, obtenhamos um vetor de *bytes* cifrado de comprimento 128, ou seja, de 1024 *bits*. Nesse projeto, o vetor de bytes na entrada é o *message digest* e temos na saída uma versão cifrada do *message digest*. Podemos também realizar a operação inversa e, a partir do dado cifrado, recuperar a informação original.

O objeto da biblioteca System.Security.Cryptography usado nesse caso foi o RSACryptoServiceProvider e os métodos foram Encrypt (.) e Decrypt (.) para a cifragem e decifragem, respectivamente.

Assinatura Digital e autenticidade

O funcionamento do sistema baseia-se no uso de uma assinatura digital gerada a partir dos dados da imagem original e nela embutida através da marca d'água. Essa imagem marcada pode, então, ser distribuída ou enviada a destinatários e usuários de acordo com a necessidade.

Assinatura digital é por definição um mecanismo que permite, de forma única e exclusiva, a comprovação da autoria ou consentimento de uma fonte em relação a determinado grupo de dados. No sistema proposto ela foi implementada através de uma função de *Hash*.

Para que obtenhamos a assinatura digital com o ferramental que dispomos, calculamos o resumo $h = H(m)$ da imagem original a ser marcada m. Em seguida, utilizamos a chave secreta do RSA, d, para calcular a assinatura $s = h^d \bmod n$.

Podemos, posteriormente, verificar a assinatura digital, bastando para isso decifra-la com a chave pública (e, n): $h = s^e \bmod n$.

Assim, a fonte ou certificadora das imagens realiza a etapa de inserção da marca com a assinatura digital e qualquer usuário da imagem pode verificar sua autenticidade bastando para isso ter recebido previamente a chave pública (e, n).

Um resumo do processo, de maneira esquemática, encontra-se nas figuras Fig. 7 e Fig. 8.

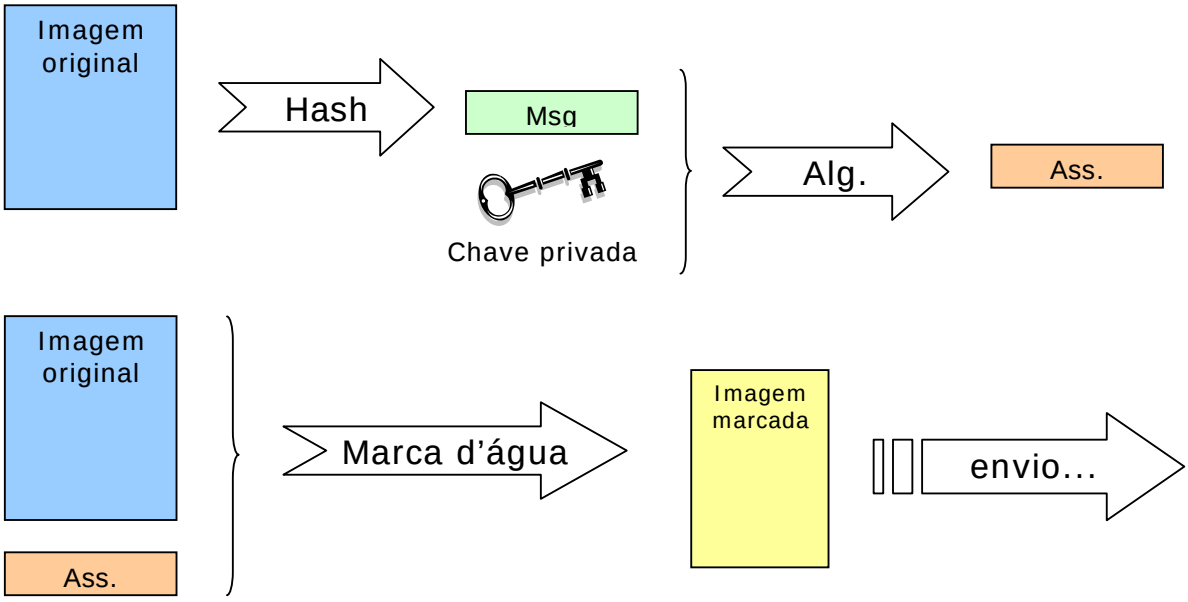

Figura 7 – Representação esquemática da elaboração da imagem marcada

A marca definida em conjunto com o algoritmo criptográfico assimétrico, a função de *Hash* e o algoritmo para (des)compactação de dados formam um único sistema, capaz de executar o mascaramento de informações e a autenticação de imagens coloridas.

Calculado o *message digest* associado à imagem, ele é cifrado por meio de um algoritmo assimétrico e de uma chave privada. O resultado desse processo é justamente a assinatura digital da imagem.

A assinatura é embutida na imagem através da técnica de marca d'água de modo a gerar a imagem marcada, que poderá ser disponibilizada a usuários e ter sua autenticidade verificada quando se desejar. A Fig. 7 representa de maneira esquemática o processo descrito.

No processo de verificação da autenticidade e/ou integridade da imagem, a imagem marcada é submetida ao processo de retirada da marca d'água ou decodificação conforme descrição em CELIK et al.'s (2005). Obtemos assim a assinatura digital contida na marca e uma nova imagem que supostamente é a imagem original. A assinatura digital pode ser decifrada por meio do algoritmo assimétrico e de uma chave pública de modo a resultar em um *message digest*.

O *message digest* dessa nova imagem é calculado e confrontado ao *message digest* proveniente da marca. Caso os dois sejam idênticos, a imagem é autêntica e, caso contrário, houve algum tipo de manipulação ou perda da integridade da imagem. A Fig. 8 representa de maneira esquemática o processo descrito.

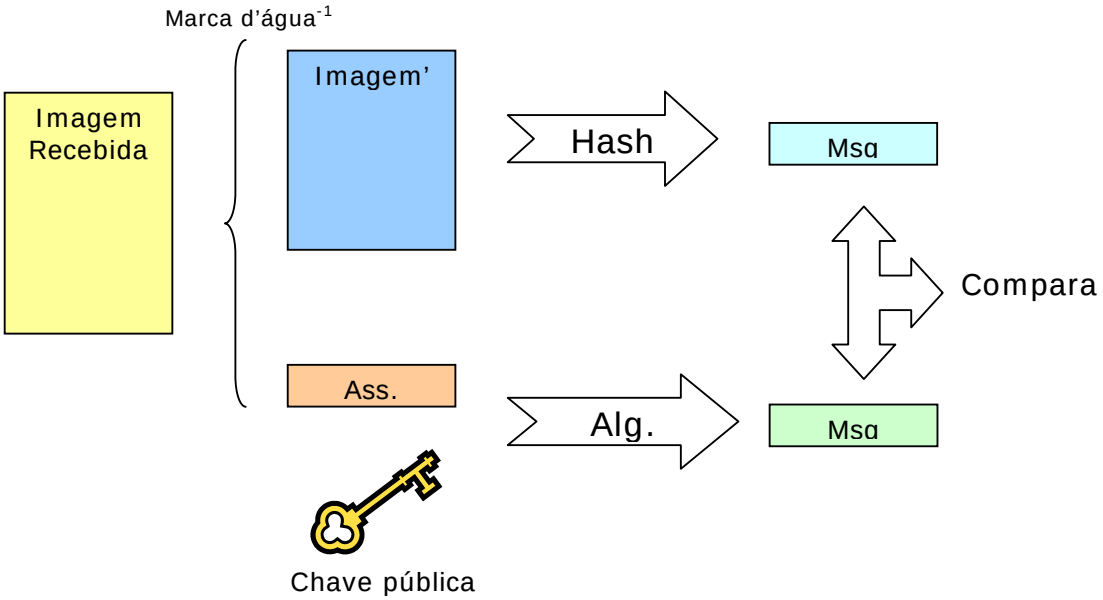

Figura 8 - Representação esquemática da verificação da autenticidade da imagem

Durante a implementação do projeto as chaves pública e privada foram atribuídas a variáveis e se mantiveram constantes (*hardcoded*). Em uma aplicação

comercial isso seria impraticável, pois tornaria inútil todo o processo de cifragem dos dados. As chaves seriam nesse caso entradas do sistema e seriam carregadas a partir de arquivos ou de uma interface homem-máquina. A estratégia adotada no projeto visou simplesmente facilitar testes e evitar erros por problemas nas chaves.

O sistema é capaz de realizar fundamentalmente três tarefas: inserir a marca d'água, verificar a autenticidade de uma imagem marcada e retirar a marca d'água reconstituindo a imagem original. Podemos com essas informações e tendo em mente que classes de usuários podem desempenhar quais tarefas, traçar um Diagrama de Casos de Uso, como é exibido na Fig. 9.

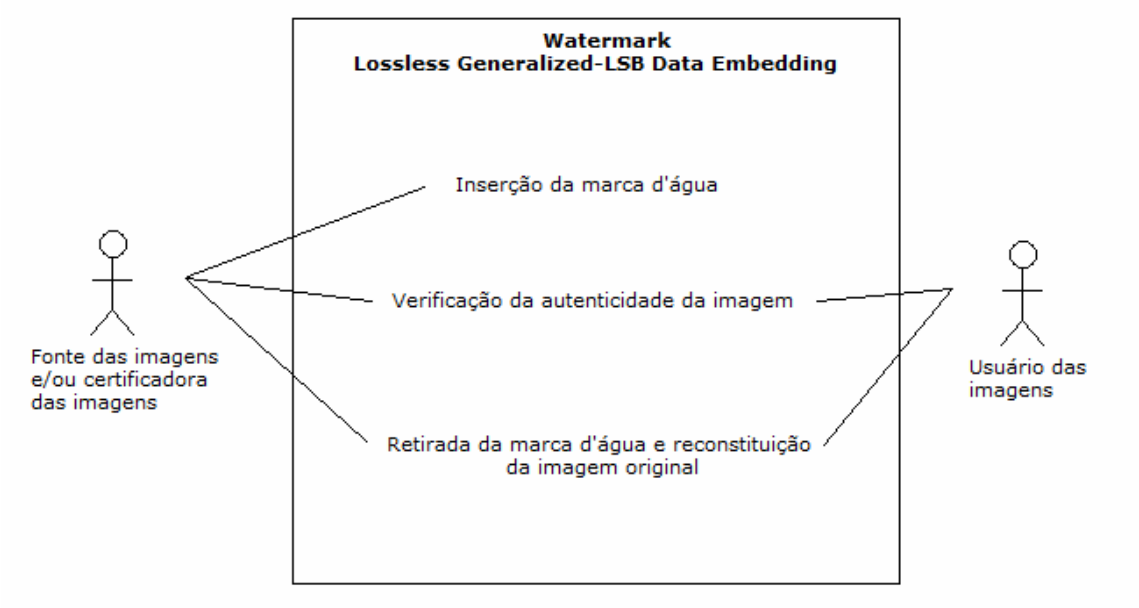

Figura 9 – Diagrama de Casos de Usos

Da modelagem da solução apresentada foi elaborado um diagrama de classes que guiou a etapa de codificação do projeto. A Fig. 10 é uma versão simplificada desse diagrama. Temos abaixo a listagem das classes implementadas com seus atributos e métodos. Em seguida há também a descrição de cada uma das classes.

Classe	Atributos	Métodos
Watermark	• string inputFile; • string outputFile; • string originalFile; • string inputMessage; • string outputMessage; • int level; • public byte colorRGB; • Boolean isAuthentic; • ImageProcessing aImageProcessing; • ByteArrayProcessing aByteArrayProcessing; • Compression aCompresser; • Hash aHash; • PublicKeyCrypto aPKC; • CALIC aCALIC; • public byte[] addressingByteArray; • public byte[] messageDigest; • public byte[] embeddedMessageDigest; • public byte[] cryptoMessageDigest; • public byte[] newImageRemainders; • public byte[] imageRemaindersCodedByCALIC; • public byte[] imageRemaindersDecodedByCALIC; • public byte[] unchangedImageRemaiders; • Watermark public byte[] level_unchangedImageRemaiders;	• public Watermark(int levelValue, string _originalFile, string _inputFile, string _outputFile) • public string Exec(string _aTask, int level) • private string Cod() • private string Decod() • public double PSNR(string originalFile, string watermarkedFile, int level, byte color) • public Boolean Authenticity()
Hash	• HashAlgorithm hash; • byte[] messageDigest;	• public byte[] CalcHashByteArray(byte[] data)
PublicKeyCrypto	• string privateKey; • string publicKey;	• public PublicKeyCrypto() • public byte[] Cod(byte[] decryptedData) • public byte[] Decod(byte[] encryptedData)

CALIC	• int level; • CALIC_ImageCodec codec;	• public CALIC(int l) • public void InitCoder(Matrix quantizedPixels, Matrix pixelsValue) • public void InitDecoder(Matrix quantizedPixels) • public byte[] Code() • public byte[] Decode() • public byte[] GetUnchangedImageRemaiders() • public Matrix GetDecodedPixelsValueMatrix() • public Boolean GetDecodeError() • public void SetUnchangedImageRemaiders(byte[] unchangedRemaiders) • public void SetCodedEmbeddedPixels(byte[] codedEmbeddedPixels)
CALIC_ImageCodec	• byte[] codedEmbeddedPixels; • byte[] unchangedImageRemaiders; • Matrix quantizedPixelsMatrix; • Matrix pixelsValueMatrix; • long countDecodedPixels; • int level; • Boolean decodeError;	• public CALIC_ImageCodec(int level) • public void InitCoder(Matrix quantizedPixels, Matrix pixelValues) • public void InitDecoder(Matrix quantizedPixels) • public byte[] Code() • public byte[] Decode() • public void SetCodedEmbeddedPixels(byte[] codedEmbeddedPixels) • public void SetUnchangedImageRemaiders(byte[] unchangedImageRemaiders) • void SetDecodeError() • public Matrix GetDecodedPixelsValueMatrix() • public byte[] GetUnchangedImageRemaiders() • public Boolean GetDecodeError() • byte LinearPrediction(int x, int y) • byte ReconstructionFunction(string position, int x, int y) • byte Delta(int x, int y, byte prediction) • byte DeltaQuantization(int delta) • short tk(byte rawPrediction, byte reconstructionFunction) • short[] tContext(int x, int y, byte prediction) • byte ContextM(int i, int j, byte prediction)

ContextTable (struct)	• short[] tContext; • int[,,,,] sumError; • int[,,,,] countError; • int[,,,,,] sumPDF; • int[,,,,] countPDF;	• public void init(int level) • public void UpdateError(short[] _tContext, byte _deltaContext, int _error) • public int AverageError(short[] _tContext, byte _deltaContext) • public void UpdatePDF(short[] _tContext, byte mContext, byte residualValue) • public double PDF(short[] _tContext, byte mContext, byte residualValue) • public double CDF_r(short[] _tContext, byte mContext, byte residualValue) • public double[] CDF(short[] _tContext, byte mContext, int level)
ArithmeticCoding	• byte[] codedByteArray; • byte[] decodedByteArray; • double[] cdf; • public long encoderCount; • public long decoderCount; • long decoderShiftsCount; • long count_S3; • double l; • double u; • double l_old; • double u_old; • double tagValue;	• public ArithmeticCoding(long codeMaxLength) • public byte[] Code(int value, double[] cdf) • public byte[] Decode(byte[] encByteArray, double[] cdf) • int numberOfBitsNeeded() • byte GetValue(double t)
ImageProcessing	• Bitmap aImage; • public int height; • public int width; • PixelFormat inputImagePixelFormat; • Matrix redImageQuantizied; • Matrix greenImageQuantizied; • Matrix blueImageQuantizied; • Matrix redImageRemainders; • Matrix greenImageRemainders; • Matrix blueImageRemainders;	• public ImageProcessing() • public Point PixelSize(Bitmap image) • public Boolean IsLoadImage24bppRgb(String _imageFileName) • public void LoadImage(String _imageFileName, int _level) • public void SaveImage(Matrix imageRemaindersMatrix, String imageFileName) • public Matrix Get_QuantiziedMatrix(string color) • public Matrix Get_ImageRemaindersMatrix(string color) • public Matrix Get_PixelsMatrix(string color)

GUI	• string inputFile; • string watermarkedFile; • string outputFile; • int level;	• public GUI() • private void GUI_Load(object sender, EventArgs e) • private void radioButton1_CheckedChanged(object sender, EventArgs e) • private void button1_Click(object sender, EventArgs e) • private void button2_Click(object sender, EventArgs e) • private void button3_Click(object sender, EventArgs e) • private void button4_Click(object sender, EventArgs e) • private void update_Form() • private void update_ErrorTextBox(string error)

Tabela 1 – Classes implementadas

A classe Watermark é o núcleo do programa, responsável pela distribuição das demandas de processamento requisitadas pela GUI. Quando o método *private void button1_Click(object sender, EventArgs e)* da GUI é disparado, os dados imputados na interface gráfica são lidos e atribuídos às variáveis correspondentes. Em seguida é chamado o *método public string Exec(string _aTask, int level)* de uma instância de objeto da classe Watermark.

Disso são disparadas todas as demandas de processamento, realizando a tarefa passada ao método *Exec* pela variável *_aTask*. Existem duas possibilidades: inserção da marca d'água ou retirada da marca d'água.

De acordo com o escolhido, a instância de Watermark realiza as chamadas necessárias às classes Hash, PublicKeyCrypto e CALIC, que respectivamente criam um *message digest* para a imagem, cifram ou decifram o *message digest* e compactam ou descompactam os dados referentes à matriz de restos da imagem.

A classe ImageProcessing é responsável por disponibilizar os métodos necessários para o processamento e síntese das imagens. Instâncias de ImageProcessing são capazes de realizar a etapa de quantização dos valores dos *pixels* de acordo com o nível de quantização *L*, carregar imagens em memória e salvar imagens.

Como dito, as classes Hash e PublicKeyCrypto juntas são responsáveis pela criação das assinaturas digitais das imagens analisadas pelo programa. Disponibilizam, de maneira transparente ao restante do *software*, os algoritmos SHA256 e RSA. Fazem uso, para isso, da biblioteca System.Security.Cryptography presente no .NET framework.

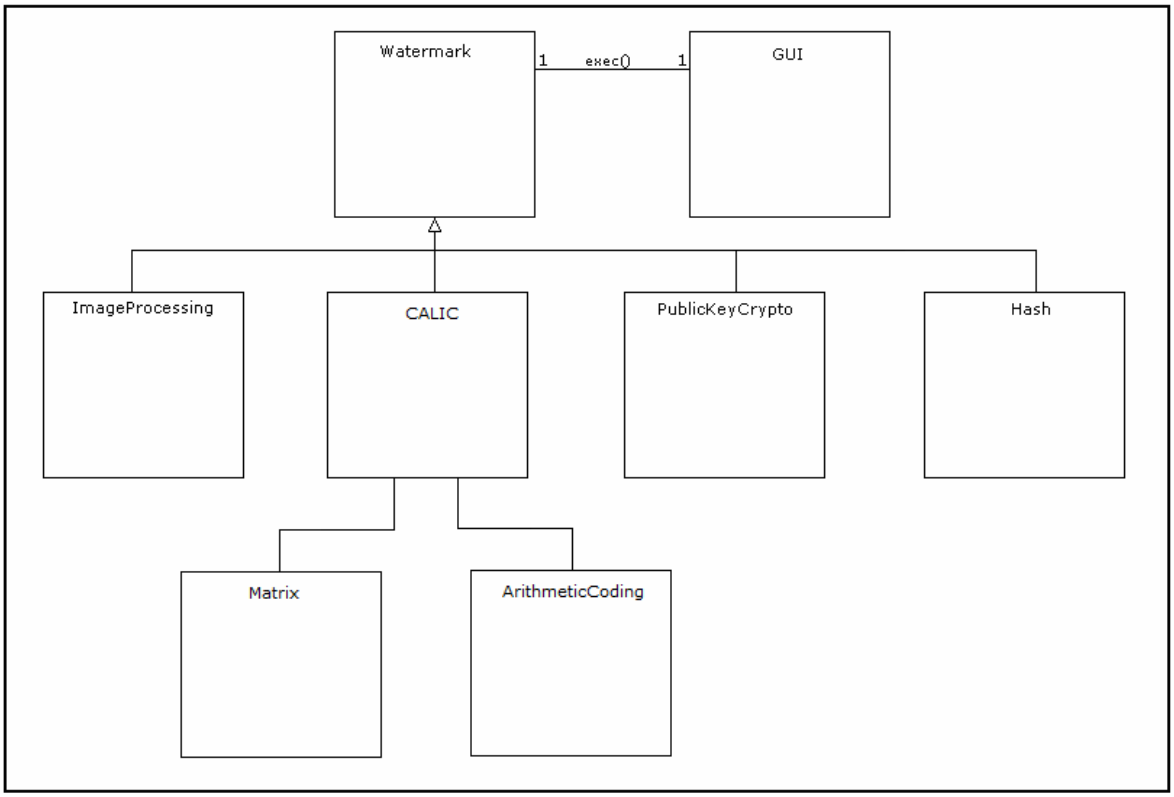

Figura 10 – Diagrama de classes simplificado

As classes CALIC e ArithmeticCoding realizam as etapas de (des)compactação dos dados embutidos na marca. Nelas foi implementado o algoritmo CALIC adaptado segundo as indicações descritas em CELIK et al.'s (2005) e um algoritmo de (de)codificação aritmética e adaptativa a contextos seguindo o descrito em SAYOOD (2000).

Duas classes foram suprimidas da Tabela 1: ByteArrayProcessing e Matrix. Essas classes permitem que os dados sejam tratados da maneira mais simples possível. Quando é mais eficiente lidarmos com vetores, utilizamos ByteArrayProcessing, quando é mais eficiente lidarmos com matrizes, utilizamos Matrix.

Em GUI encontra-se toda a implementação dos atributos e métodos necessários à interface homem-máquina. Na figura Fig. 11 e Fig. 12 temos uma visão dessa interface.

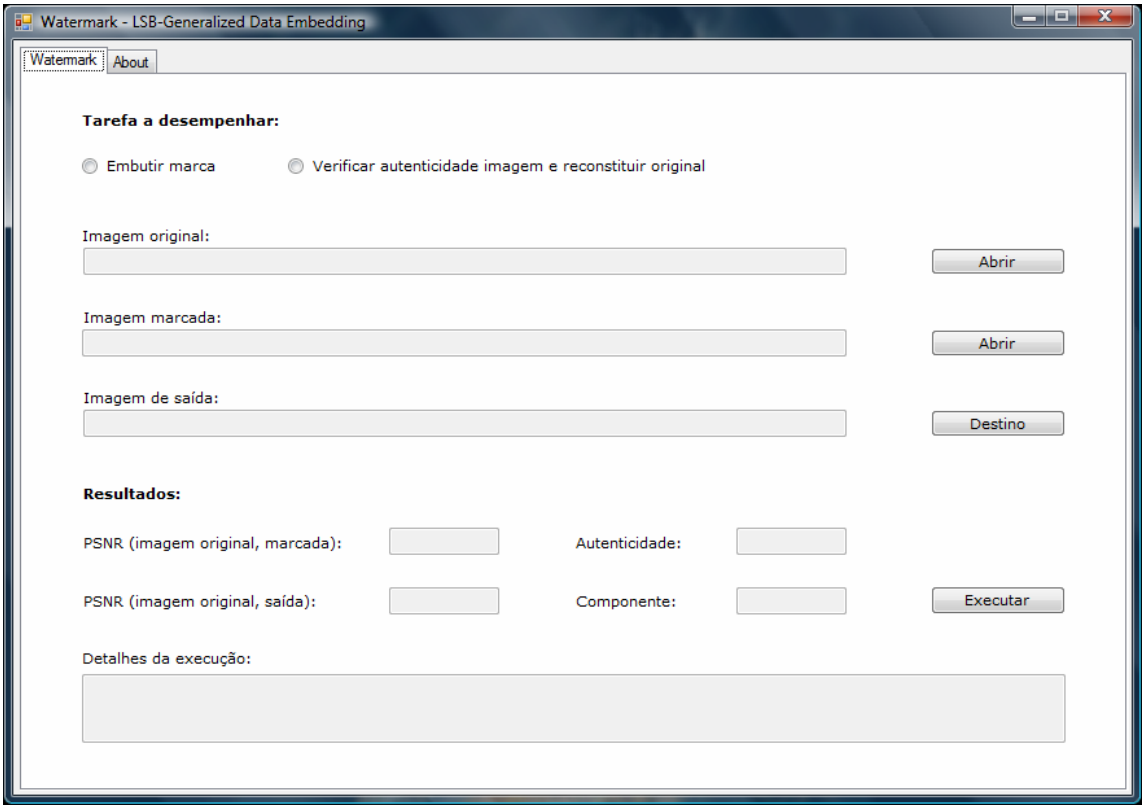

Figura 11 - Interface gráfica 1

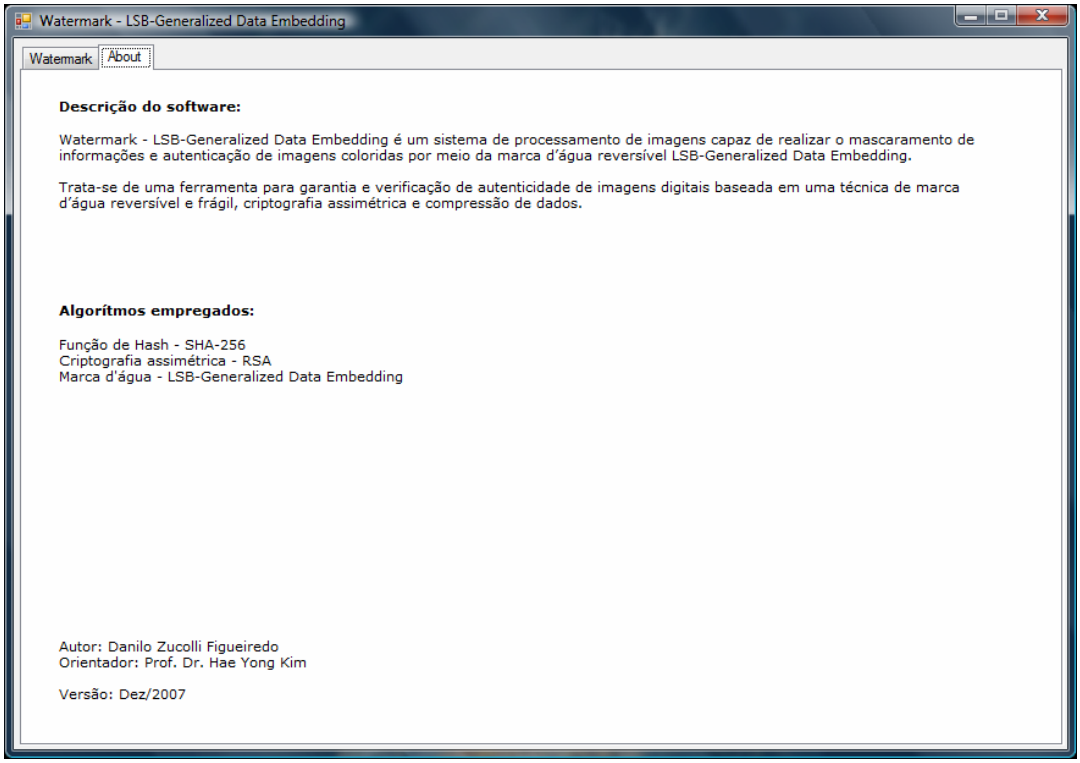

Figura 12 - Interface gráfica 2

Conforme podemos observar a Interface apresenta duas abas. A primeira tem como função permitir a entrada dos dados e exibir os resultados da execução, já a segunda tem como função transmitir ao usuário algumas informações sobre como o *software* foi implementado.

As entradas de dados são: a tarefa que se deseja executar e as imagens de entrada e saída. Quando se deseja embutir uma marca, deve-se entrar com a imagem original e a imagem de saída, que nesse caso será a imagem marcada. Já quando se deseja verificar a autenticidade de uma imagem marcada e reconstituir a imagem original, deve-se entrar com a imagem marcada e a imagem de saída, que nesse caso será a imagem original reconstituída.

Existe também na interface uma porção para exibição de resultados. Ali há o resultado dos cálculos de valor de PSNR entre as imagens envolvidas no processamento e uma parte destinada à exibição de detalhes da execução, como erros ou sucesso no processamento.

5 RESULTADOS

O resultado do projeto foi a elaboração de um sistema de processamento de imagens que permite a realização do mascaramento de informações e autenticação de imagens coloridas por marcas d'água reversíveis.

O sistema é capaz de realizar a inserção e extração da marca d'água, verificação da autenticidade da imagem marcada e reconstituição da imagem original, eliminando-se a marca, além da cifragem e decifragem da mensagem transmitida na marca.

A especificação técnica do projeto foi estabelecida em termos de PSNR e impôs que as imagens marcadas tivessem um valor mínimo de PSNR igual a 30dB.

Durante os diversos testes do projeto realizados em imagens coloridas do tipo BMP, bitmap de 24 *bits* por *pixel*, constatou-se que a meta técnica do projeto foi satisfeita.

É evidente o compromisso entre qualidade da imagem marcada e quantidade de dados embutidos, sendo os dois inversamente proporcionais. Para que esse compromisso de projeto fosse adequadamente tratado, adotamos arbitrariamente a estratégia de inserir a marca d'água na componente *Red* das intensidades de cores dos *pixels* da imagem.

Além disso, os dados embutidos foram apenas a assinatura digital da imagem, os restos da quantização dos dados e alguns sinais para o controle das etapas de cifragem e decifragem das imagens.

Por isso, pôde-se adotar um baixo nível de quantização dos dados de intensidade das cores das imagens. O valor definido foi $L = 4$, que foi suficiente para a inserção dos dados nas figuras testadas.

Foi realizada ainda uma implementação alternativa, na qual foi utilizado o algoritmo Deflate na etapa de (des)compressão de dados. Essa solução alternativa mostrou-se viável para imagens coloridas em tons discretos, mas mostrou-se também inviável para imagens em tons contínuos.

Imagens em tons discretos tiveram resultados em termos de PSNR da ordem de 40 dB, enquanto imagens em tons contínuos não puderam ser marcadas, pois o algoritmo Deflate não foi capaz de realizar a compactação dos dados. Essa solução

alternativa, implementada meramente para estudo e testes, demonstrou a necessidade do algoritmo CALIC adaptado.

Implementada a versão completa da solução, já contando com a assinatura digital (via SHA-256 / RSA) e com a compactação de imagens feita por meio do algoritmo CALIC adaptado, os resultados demonstraram a viabilidade da ‘solução tanto para imagens em tons discretos quanto em tons contínuos.

Diferentemente da solução que usou o algoritmo Deflate para a etapa de compactação dos dados, a compactação dos dados se tornou viável nessa versão graças ao algoritmo CALIC adaptado.

Uma série de imagens de diferentes tamanhos de área e dispersões de cores foi testada e os resultados foram imagens marcadas com qualidade superior aos 30 dB de PSNR que era a meta técnica do projeto. A média dos valores de PSNR para o universo das imagens testadas foi superior a 40 dB. Abaixo temos alguns exemplos de imagens testadas.

Figura 13 - Bob Marley.bmp

Figura 14 - Bonecos.bmp

Figura 15 - Cat.bmp

Figura 16 - Easter island.bmp

Figura 17 - Família.bmp

Figura 18 - Gourd drum.bmp

Figura 19 - Jardim.bmp

Figura 20 - Lena.bmp

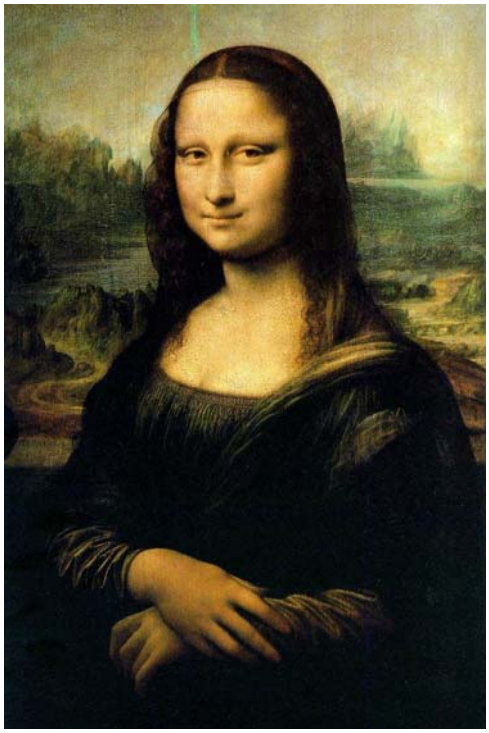

Figura 21 - Monalisa.bmp

Figura 22 - Papagaios.bmp

Figura 23 - Pipoca.bmp

Figura 24 - Van Gogh.bmp

Na tabela Tab.1 estão resumidos os resultados obtidos. Nela os tempos de processamento se referem ao tempo para a execução da inserção da marca. O tempo para extração da marca possui comportamento semelhante.

Imagem	Dimensões (pixels x pixels)	Tempo de Processamento (s)	PSNR (dB)
Bob Marley	1280 x 1024	8,69	44,03
Bonecos	683 x 1024	4,07	43,84
Cat	1600 x 1200	10,81	43,97
Easter Island	1600 x 1200	10,82	44,12
Família	1632 x 1224	11,01	44,12
Gourd drum	992 x 1191	6,84	44,05
Jardim	1024 x 768	4,52	43,02
Lena	512 x 512	2,01	44,08
Monalisa	740 x 1113	5,38	44,10
Papagaios	192 x 128	0,57	44,11
Pipoca	2592 x 1728	29,83	43,85
Van Gogh	1061 x 808	5,26	44,13

Tabela 2 – Tempo de processamento e PSNR

Podemos, a partir dos dados obtidos, analisar o tempo de processamento do software em função da variação das áreas das imagens processadas. Plotando os dados em um gráfico de tempo de processamento, em segundos, em função da área, em pixels^2, temos:

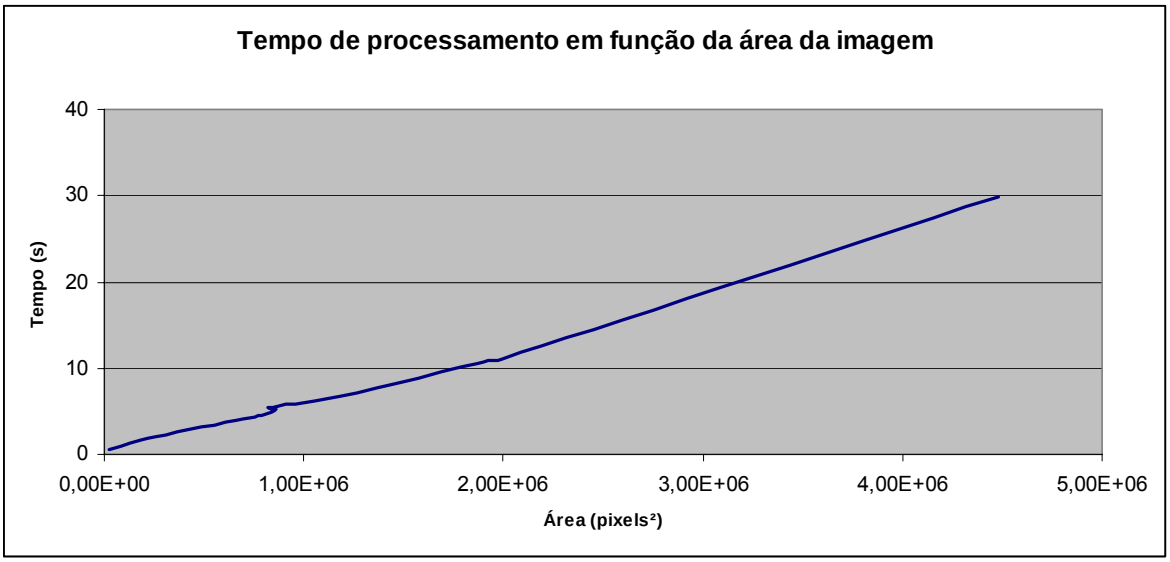

Gráfico 1 – Tempo de processamento em função da área da imagem

A curva resultante tem forma que se aproxima muito a uma reta, demonstrando que o processamento *pixel-a-pixel* do *software* implica em um aumento linear no tempo de processamento em função do aumento de área da imagem.

Outra análise interessante é observar como varia a qualidade das imagens marcadas em função do nível de quantização, *L*, a que são submetidas. Um valor de *L* crescente implica em um maior número de bits menos significativos disponíveis para a inserção da marca d'água, ou seja, em um maior número de bits potencialmente modificados.

L	**PSNR**		
	Papagaios.bmp	**Lena.bmp**	**Pipoca.bmp**
2	~ ∞	51,14	51,12
4	44,11	44,08	43,85
8	37,66	37,64	37,27
16	31,36	31,11	30,51
32	24,49	24,40	24,68

Tabela 3 – Análise de PSNR em função do nível de quantização

É notória a crescente degradação das imagens marcadas em função do valor crescente do nível de quantização, *L*. Notamos também que esse fato é independente das dimensões das imagens, uma vez que as três imagens analisadas apresentam dimensões significativamente diferentes, mas resultados muito semelhantes.

6 DISCUSSÃO

Neste projeto foi desenvolvida uma implementação de técnica de mascaramento de informações e autenticação de imagens coloridas por marcas d'água reversíveis para o caso de imagens coloridas não-compactadas, tendo sido elaborado um protótipo de *software* que implementa a função de autenticação proposta fazendo uso de um algoritmo de criptografia assimétrico.

A técnica de marca d'água utilizada foi a *Lossless Generalized-LSB Data Embedding*, descrita em CELIK et al. (2005). Trata-se de uma técnica recente e que faz parte do Estado da Arte na área de Marcas d'água frágeis e reversíveis, servindo de base para implementações de sistemas de autenticação de dados digitais e para novos desenvolvimentos na área.

Esta técnica inclui uma etapa que envolve (des)compressão de dados, onde é aplicado uma adaptação do algoritmo CALIC. Essa adaptação foi feita para que as informações disponíveis durante o processamento da imagem – principalmente a matriz dos valores quantizados - fossem utilizadas ao máximo, de modo a otimizar a compressão dos dados.

O CALIC, Context-Based, Adaptive, Lossless Image Coding, é um algoritmo que realize a (des)compressão sem perdas de imagens digitais. Para isso cria um modelo para a imagem, no qual um preditor é implementado e erros de predição são calculados. Isso junto à modelagem dos contextos da imagem permite a um algoritmo de codificação aritmética realizar a compressão dos dados.

O algoritmo de criptografia utilizado foi o Rivest, Shamir, Adleman (RSA), que apesar de não ser mais considerado o Estado da Arte em termos de algoritmos criptográficos assimétricos, é o mais conhecido e aplicado método criptográfico desse tipo.

Foi utilizado também o SHA-256, que é uma função de resumo criptográfico vastamente empregada em aplicações que mantém relação com autenticação de dados. Esse algoritmo é ainda considerado seguro, diferentemente do SHA-1, satisfazendo as três propriedades que definem uma função de *Hash*.

Utilizados em conjunto, o RSA e o SHA-256 produziram a assinatura digital utilizada para a autenticação das imagens. Além disso, a assinatura digital foi o dado

embutido nas imagens, pois as demais informações embutidas tinham apenas o propósito de permitir que a marca fosse reversível.

A assinatura digital e a marca d'água formaram um sistema completo para a autenticação de imagens digitais.

O protótipo desenvolvido, por construção, atendeu tanto à funcionalidade proposta quanto à meta técnica de imagens marcadas com valor mínimo de PSNR igual a 30dB. Analisando os resultados pode-se observar que a meta de 30dB foi superada e os resultados chegaram a um patamar de PSNR médio de 40dB.

7 CONCLUSÕES

Conforme descrito, neste projeto foi desenvolvida uma implementação de técnica de mascaramento de informações e autenticação de imagens coloridas por marcas d'água reversíveis para o caso de imagens coloridas não-compactadas, tendo sido elaborado um protótipo de *software* que implementa a função de autenticação proposta fazendo uso de um algoritmo de criptografia assimétrico.

Para isso todo um ferramental algorítmico teve que ser analisado e implementado. Dentre os elementos que compuseram a solução estão os algoritmos SHA 256, RSA e a marca d'água digital *Lossless Generalized-LSB Data Embedding*.

A implementação obtida foi bem sucedida, uma vez que o protótipo de *software* elaborado é capaz de realizar a autenticação de imagens digitais atendendo as metas técnicas propostas durante a especificação do projeto.

A abordagem utilizada durante a fase de desenvolvimento e codificação foi extremamente pragmática e contou com uma série de instrumentos de apoio. Dentre eles destacam-se os elementos contidos no ambiente de desenvolvimento Microsoft Visual C# 2005 Express Edition e no .NET framework.

A aplicação obtida enquadra-se no conjunto de soluções para questões como propriedade, integridade e autenticação de dados que se tornou fundamental com o intenso compartilhamento de dados via redes de computadores e sua utilidade é imensa.

BIBLIOGRAFIA

ALATTAR, A. M. Reversible watermark using the difference expansion of a generalized integer trans-form. IEEE Transactions on Image Processing, vol. 13, no. 8, p. 1147–1156, Agosto 2004.

CELIK, M. U.; SHARMA G.; TEKALP, A. M. e SABER E. Lossless generalized-lsb data embedding. IEEE Transactions on Image Processing, vol. 14, no. 2, p. 253–266, Fevereiro 2005.

CHANG, C. C.; TAI, W. L. e LIN M. H. A reversible data hiding scheme with modified side match vector quantization. Proceedings of the International Conference on Advanced Information Networking and Applications, vol. 1, p. 947–952, Taiwan, Março 2005.

DAHAB R. e LOPEZ J.C., Técnicas criptográficas modernas: algoritmos e protocolos. Instituto de Computação – UNICAMP. Abril 2007.

FENG J.B. et al. Reversible Watermarking: Current Status and Key Issues. International Journal of Network Security, vol. 2, no 3, p.161–171, Maio 2006.

FRIDRICH J., GOLJAN M. e DU R., Lossless data embedding-new paradigm in digital
watermarking. EURASIP Journal of Signal Processing, vol. 2002, no. 2, p. 185–196, Fevereiro 2002.

HONSINGER, C. W.; JONES, P.; RABBANI, M. e STOFFEL, J. C. Lossless recovery of an original image containing embedded data. US Patent, vol. 6, 278,791, 2001.

KALKER, T. e WILLEMS, F. M. Capacity bounds and code constructions for reversible data-hiding. Proceedings of Electronic Imaging 2003, Security and Watermarking of Multimedia Contents V, p. 604–611, EUA, Janeiro 2003.

KIM, H. Y. e AFIF, A. Secure Authentication Watermarking for Binary Images. Proc. Sibgrapi - Brazilian Symp. on Comp. Graph. and Image Proc., p. 199-206, 2003.

LEEST, A.; VEEN, M. e BRUEKERS, F. Reversible image watermarking. Proceedings of the ICIP International Conference on Image Processing, vol. 3, p. II-731-4, Espanha, Setembro 2003.

NI, Z.; SHI, Y. Q.; ANSARI, N. e WEI, S. Reversible data hiding, ISCAS Proceedings of the 2003 International Symposium on Circuits and Systems, vol. 2, p. II–912–II–915, Thailand, Maio 2003.

SAYOOD, K. Introduction to Data Compression. Second Edition. The Morgan Kaufmann Series in Multimedia Information and Systems. Morgan Kaufmann Publishers. 2000.

THODI, D. M. e RODRIGUEZ, J. J. Reversible watermarking by prediction-error expansion. Proceedings of the 6th IEEE Southwest Symposium on Image Analysis and Interpretation, vol. 3, p. 21–25, Lake Tahoe, EUA, Março 2004.

TIAN J. Reversible data embedding using a difference expansion. IEEE Transactions on Circuits Systems and Video Technology, vol. 13, no. 8, p. 890– 896, Agosto 2003.

TIAN, J. Wavelet-based reversible watermarking for authentication. Proceedings of SPIE Sec. and Watermarking of Multimedia Cont. IV, vol. 4675, Janeiro 2002.

TSAI, C. L.; FAN, K. C.; CHUNG, C. D. e CHUNG, T. C. Reversible and lossless data hiding with application in digital library. Proceedings of the 38th Annual International Carnahan Conference on Security Technology, p. 226–232, EUA, Outubro 2004.

VLEESCHOUWER, C. D.; DELAIGLE, J. F. e MACQ, B. Circular interpretation of bijective transformations in lossless watermarking for media asset management. IEEE Transactions on Multimedia, vol. 5, no. 1, p. 97–105, Março 2003.

VLEESCHOUWER C. D.; DELAIGLE J. E. e MACQ B., Circular interpretation of histogram for reversible watermarking. Proceedings of the IEEE 4th Workshop on Multimedia Signal Processing, p. 345–350, França, Outubro 2001.

XUAN, G.; YANG, C.; ZHEN, Y.; SHI, Y. Q. e NI, Z. Reversible data hiding based on wavelet spread spectrum. Proceedings of the IEEE 6th Workshop on Multimedia Signal Processing, p. 211–214, Itália, Setembro 2004.

WANG X., YIN Y.L. e YU H. Collision Search Attacks on SHA1. Fevereiro 13, 2005

WU X. e Memon N. Context-Based, Adaptive, Lossless Image Coding. IEEE Transactions on Communications, vol. 45, no. 4, p. 437–444. Abril 1997.

YANG, B.; SCHMUCKER, M.; NIU, X.; BUSCH C. e SUN, S. Reversible image watermarking by histogram modification for integer dct coefficients. Proceedingd of the IEEE 6th Workshop on Multimedia Signal Processing, p. 143–146, Itália, Setembro 2004.

ZAIN, J.M.; BALDWIN, L. P. e CLARKE, M. Reversible watermarking for authentication of dicom images. Proceedings of the 26th Annual International Conference of the Engineering in Medicine and Biology Society, p. 3237–3240, EUA, 2004.

ZHANG, L. B. e WANG, K. Embedded multiple sub-bands scaling image coding using reversible integer wavelet transforms. Proceedings of the International Symposium on Intelligent Multimedia, Video and Speech Processing, p. 599–602, Hong Kong, Outubro 2004.