Marcas d'Água Frágeis de Autenticação para Imagens em Tonalidade Contínua e Esteganografia para Imagens Binárias e Meio-Tom

Hae Yong Kim[1]

Resumo: Uma marca d'água é um sinal portador de informação acrescentada ao dado digital que pode ser extraída mais tarde para fazer alguma asserção sobre o dado. As marcas d'água digitais são normalmente classificadas em robustas e frágeis. As marcas robustas são projetadas para resistirem a maioria dos procedimentos de manipulação de imagens e normalmente são usadas para atestar a propriedade da imagem. As marcas frágeis são facilmente corrompidas por qualquer processamento na imagem. Porém, as marcas para checar integridade da imagem devem ser frágeis, para que qualquer alteração seja detectada. Este tutorial irá expor as principais técnicas de marcas d'água frágeis para imagens estáticas de tonalidade contínua. Também serão expostas as principais ataques contra estas marcas d'águas e os meios para se defender. Em seguida, irá expor as principais técnicas para embutir dados em imagens binárias. Restringiremos o estudo somente para as imagens não-compactadas.

Palavras-chave: Marca d'Água Frágil, Esteganografia, Meio-tom.

Abstract: Watermark is an information-carrying signal embedded into digital data that can be extracted later to make some assertion about the data. Digital watermarks are usually classified as robust or fragile. Robust watermarks are designed to resist most image manipulating procedures and are used normally to attest the image ownership. Fragile watermarks are easily corrupted by any image processing procedure. However, integrity checking watermarks must be fragile, in order to detect any alteration. This tutorial course expounds principal authentication watermarking techniques for still continuous-tone images. It also expounds major attacks against these techniques and how to defend against them. Afterwards, this tutorial will describe major techniques to embed data into binary images. We will restrict our study to non-compressed images only.

Keywords: Fragile Watermarking, Steganography, Halftoning.

1 Escola Politécnica, USP, Av. Prof. Luciano Gualberto, tr. 3, 158, 05508-900, São Paulo, SP.
{hae@lps.usp.br}, http://www.lps.usp.br/~hae

1 Introdução

O espetacular crescimento de sistemas de multimídia interligados pela rede de computadores nos últimos anos (particularmente com o advento do World Wide Web) tem apresentado um enorme desafio nos aspectos tais como propriedade, integridade e autenticação de dados digitais (áudio, vídeo e imagens estáticas). Para enfrentar tal desafio, o conceito de marca d'água digital foi definido.

Uma marca d'água é um sinal portador de informação, visualmente imperceptível, embutido numa imagem digital. Quando não houver perigo de confusão, utilizaremos a palavra "marca" como sinônimo de "marca d'água".

A imagem que contém uma marca d'água é dita imagem marcada ou hospedeira. Apesar de muitas técnicas de marca d'água poderem ser aplicadas diretamente para diferentes tipos de dados digitais, este tutorial focalizará a atenção somente em marcas para imagens digitais 2-D estáticas.

O primeiro degrau no estudo das marcas d'água é o estudo das técnicas utilizadas para esconder a informação numa imagem, conhecida como esteganografia (information hiding ou steganography em inglês). Nesta área de pesquisa, estuda-se como inserir a maior quantidade possível de informações com uma mínima deterioração na qualidade da imagem hospedeira, sem estar preocupado com a finalidade ou utilidade da informação escondida ou se a informação escondida é fácil ou difícil de ser removida. Dificuldades especiais para inserir dados escondidos aparecem quando se utilizam formatos de imagens compactadas com perdas.

As marcas d'água digitais são classificadas de acordo com a dificuldade em removê-las em robustas, frágeis e semifrágeis. Esta classificação também normalmente determina a finalidade para qual a marca d'água vai ser utilizada.

As marcas robustas são projetadas para resistirem a maioria dos procedimentos de manipulação de imagens. A informação embutida numa imagem através de uma marca d'água robusta deveria ser possível de ser extraída mesmo que a imagem hospedeira sofra rotação, mudança de escala, mudança de brilho/contraste, compressão com perdas com diferentes níveis de compressão, corte das bordas (cropping), etc. Uma boa marca d'água robusta deveria ser impossível de ser removida a não ser que a qualidade da imagem resultante deteriore a ponto de destruir o seu conteúdo visual. Isto é, a correlação entre uma imagem marcada e a marca d'água robusta nela inserida deve permanecer detectável mesmo após um processamento digital, enquanto a imagem resultante do processamento continuar visualmente reconhecível e identificável como a imagem original. Por esse motivo, marcas d'água robustas são normalmente utilizadas para a verificação de propriedade ou de *copyright* das imagens. Para dar um exemplo, se uma agência de notícias colocasse uma marca robusta numa fotografia, nenhum adulterador malicioso deveria ser capaz de remover essa marca.

Apesar de muitas pesquisas, parece que ainda não foi possível obter uma marca d'água robusta realmente segura.

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

As marcas d'água frágeis de autenticação detectam *qualquer* alteração na imagem. Às vezes, esta propriedade é indesejável. Por exemplo, ajustar brilho/contraste para melhorar a qualidade da imagem pode ser um processamento válido, que não deveria ser detectada como uma tentativa de adulteração maliciosa. Ou então, compactar uma imagem com perdas (como JPEG ou JPEG2000) em diferentes níveis de compressão deveria ser uma operação permitida. Ainda, imprimir e escanear uma imagem não deveria levar à perda da autenticação. Assim, foram criadas as marcas d'água semifrágeis. Uma marca d'água semifrágil também serve para autenticar a imagem. Só que estas procuram distinguir as alterações que modificam uma imagem substancialmente das alterações que não modificam o conteúdo visual da imagem. Elas normalmente extraem algumas características de imagem que permanecem invariantes através das operações "permitidas" e as insere de volta na imagem de tal forma que a alteração de uma dessas características possa ser detectada.

Podemos subdividir as marcas d'água de autenticação (tanto frágeis como semifrágeis) em três subcategorias: sem chave, com chave secreta (cifra simétrica) e com chave pública/privada (cifra assimétrica):

Uma marca de autenticação sem chave é útil para detectar alterações não-intencionais na imagem tais como erro de transmissão ou de armazenamento. Funciona como uma espécie de "check-sum". Se um algoritmo de autenticação sem chave estiver disponível publicamente, qualquer pessoa pode inserir este tipo de marca em qualquer imagem e qualquer pessoa verificar se uma imagem contém uma marca.

A marca de autenticação com chave secreta (cifra simétrica) é usada para detectar qualquer alteração na imagem, mesmo uma alteração intencional ou maliciosa. Este tipo de marca é similar aos códigos de autenticação de mensagem, sendo que a única diferença é que o código de autenticação é inserido na imagem em vez de ser armazenada separadamente. Algoritmos para inserção e detecção deste tipo de marca podem ser disponibilizados publicamente, e uma chave secreta é usada em ambas fases. Vamos supor que Alice administra um grande banco de dados de imagens, onde cada imagem está assinada com uma chave secreta k que somente Alice conhece. Vamos supor que Mallory, um hacker malicioso, modifique uma imagem neste banco de dados. Mallory não consegue inserir a marca correta na imagem adulterada pois ele não conhece a chave k. Mais, Alice será capaz de detectar todas as imagens alteradas pelo Mallory usando o algoritmo de detecção de marca d'água e sua chave secreta k.

As marcas de autenticação com chave pública (cifra assimétrica) utilizam criptografia de chave pública para inserir uma assinatura digital na imagem. Usando cifras de chave pública, a autenticidade de uma imagem pode ser julgada sem a necessidade de tornar pública qualquer informação privada. Os possíveis usos deste tipo de marca d'água são muito variados:

Vamos supor que uma agência de notícias chamada Alice deseja distribuir pela internet uma fotografia jornalística, com alguma prova de autenticidade de que a foto foi distribuída pela Alice e que ninguém introduziu alterações maliciosas na foto. Alice utiliza a sua chave privada para inserir marca d'água de autenticação na imagem e distribui a foto marcada. Vamos supor que Bob recebe a foto marcada. Bob usa a chave pública de Alice para verificar que a foto está assinada pela Alice e que ninguém introduziu qualquer alteração depois de Alice assiná-la. Se Mallory, um hacker malicioso, alterar a foto, ele não poderá inserir a marca correta na imagem falsificada porque ele não conhece a chave privada da Alice. Além disso, Mallory não poderá distribuir uma foto como sendo Alice porque ele não conseguirá assiná-la por desconhecer a chave privada da Alice.

Marca d'água de autenticação de chave pública pode ser usada, por exemplo, em "câmera digital confiável". Se uma câmera digital possuir internamente uma chave privada e assinar todas as fotos tiradas por ela com uma marca d'água utilizando a sua chave secreta, as pessoas poderão ter certeza de que uma foto foi tirada com uma determinada máquina utilizando a correspondente chave pública. Se a foto fosse adulterada, a marca d'água não poderia mais ser verificada.

De forma semelhante, uma "máquina de FAX confiável" poderia conter internamente uma chave privada e inserir uma marca d'água em todos os documentos transmitidos por ela. O receptor de FAX, usando a chave pública da máquina transmissora, pode verificar que o documento foi originado de uma máquina de FAX específica e que o documento não foi manipulado.

Este tutorial está dividida em três grandes unidades.

Na primeira unidade (seção 2), apresentaremos os conceitos básicos necessários para o restante do curso: assinatura digital e algoritmo de meio-tom pela difusão de erro.

A segunda unidade (seção 3) irá expor as principais marcas d'água de autenticação para imagens de tonalidades contínuas (tanto em níveis de cinza como coloridas). As técnicas de marcas de autenticação para imagens de tonalidades contínuas já estão bastante desenvolvidas. Também serão expostas as principais ataques contra as marcas d'águas de autenticação e os meios para se defender contra tais ataques. Trataremos apenas marcas d'água para formatos de imagens sem perdas.

Na terceira unidade (seção 4), este tutorial apresentará as principais técnicas para embutir dados escondidos (esteganografia) em imagens binárias (tanto binárias simples como imagens meio-tom). A esteganografia é um estágio anterior à obtenção das marcas d'água frágeis e/ou robustas. As marcas d'água frágeis ou robustas fazem uso das técnicas de esteganografia para embutirem dados. Aparentemente, as marcas d'água de autenticação para

imagens binárias não se encontram num estágio tão desenvolvido como para as imagens de tonalidades contínuas. Assim, estudaremos as técnicas disponíveis no momento, que somente embutem dados em imagens binárias sem que de fato autentiquem a imagem.

Marca d'água e esteganografia é uma área de pesquisa extremamente ativa, e novas técnicas aparecem com grande freqüência. A finalidade deste tutorial é simplesmente dar um vislumbre de algumas das técnicas que estão sendo pesquisadas, sem ter a pretensão de esgotar o assunto.

2 Conceitos Preliminares

Iniciaremos este tutorial apresentando dois conceitos que serão utilizados ao longo deste curso: assinatura digital e meio-tom (halftone) por difusão de erro.

2.1 Assinatura digital

Para apresentar o conceito de assinatura digital, seguiremos de perto a redação didática de [Friedman, 1993] e [Barreto, 2003].

A criptografia de chave secreta ou simétrica requer que tanto o transmissor como o receptor de mensagem possuam a mesma chave secreta: o transmissor utiliza a chave para transformar a mensagem original em texto cifrado, e o receptor utiliza a mesma chave para executar a transformação inversa, recuperando o texto original. O defeito histórico deste esquema é a distribuição segura das chaves: a chave deve ser transmitida através de um caro meio seguro alternativo.

O conceito de criptografia de chave pública foi inventado pelo W. Diffie e M. Hellman, e independentemente por R. Merkle [Schneier, 1996, chap. 19]. Criptografia de chave pública ou cifra assimétrica utiliza duas chaves: uma chave privada e outra chave pública. Conhecendo a chave privada, é fácil e rápido calcular a chave pública correspondente. Porém, o contrário é uma tarefa extremamente difícil computacionalmente (levaria talvez séculos utilizando os supercomputadores atuais).

Para enviar uma mensagem secreta que somente o receptor Bob possa ler, Bob primeiro torna conhecida publicamente a sua chave pública. Qualquer pessoa que queira enviar uma mensagem secreta a Bob deve criptografar a mensagem usando esta chave pública e enviá-la a Bob. Bob, sendo único possuidor da chave secreta, é a única pessoa capaz de decriptografar a mensagem. Note que a necessidade de combinar uma chave secreta entre transmissor e receptor foi eliminada.

O processo descrito acima pode ser implementado "ao contrário". Neste caso, a transmissora Alice é que guarda uma chave privada, e a chave pública é disponibilizada publicamente a qualquer receptor que queira decriptografar. Este procedimento não mais executa a função tradicional de criptografia, que é permitir uma comunicação confidencial entre duas partes. Porém, fornece um meio para assegurar que as mensagens não foram forjadas: somente Alice, que possui a chave privada, poderia ter produzido uma mensagem que é decifrável pela correspondente chave pública.

As assinaturas digitais estão construídas sobre as técnicas de criptografia de chave pública. Elas permitem autenticar o conteúdo da mensagem e a identidade do emissor.

As assinaturas são produzidas através de uma função *hash*. Intuitivamente, uma função *hash* calcula, rápida, segura e univocamente, representantes adequadamente curtos para mensagens arbitrariamente longas (chamadas "impressões digitais" das mensagens).

Essas "impressões digitais" são criptografadas utilizando a chave privada, em lugar das próprias mensagens. Isto acelera tanto o processo de criar assinatura como o processo de verificar a assinatura. O resultado é um dado (chamado de assinatura digital) que acompanha a mensagem original. Desta forma, a mensagem original pode ser lida por todos, porém se um receptor chamado Bob deseja autenticá-la, Bob pode decriptografar a assinatura digital da mensagem usando a chave pública, obtendo a "impressão digital" da mensagem. Esta impressão digital deve ser idêntica a o *hash* da mensagem original, se a mensagem não tiver sido adulterada.

Uma assinatura digital diz-se *determinística* se o seu valor exclusiva e univocamente determinado pelos dados assinados e pela chave privada do signatário. Em contraste, uma assinatura digital é *não-determinística* se depende também de algum parâmetro aleatório, chamado *sal* ou *nonce*. Assume-se que esse parâmetro seja estatisticamente único (irrepetível) e uniformemente distribuído. Além disso, o seu valor deve ser imprevisível para um adversário do sistema. Alguns esquemas de assinatura (por exemplo, DSA e Schnorr [Schneier, 1996]) são naturalmente não-determinísticos. Outros esquemas (por exemplo, RSA) precisam de construções especiais para que se tornem não-determinísticos.

Descreveremos resumidamente, aquele que é provavelmente o algoritmo de criptografia de chave pública mais amplamente utilizado atualmente, o algoritmo RSA (Rivest, Shamir e Adleman [Rivest, 1978]). Sejam p e q dois números primos distintos de tamanhos aproximadamente iguais, seja $n = pq$, e seja e um inteiro inversível módulo $(p-1)(q-1)$, com inverso $d \equiv e^{-1} \bmod (p-1)(q-1)$ (isto é, $ed \equiv 1 \pmod{(p-1)(q-1)}$). A chave pública é o par (e, n), e a chave privada é o inteiro d (os primos p e q são também mantidos secretos, e podem até ser descartados, pois o conhecimento deles não é essencial para operações RSA). Seja $M \in \mathbb{Z}_n$ a mensagem a ser assinada. Uma assinatura RSA para M é definida como $C = M^d \bmod n$. A verificação da assinatura procede com a recuperação de M a partir de C: $M = C^e \bmod n$.

2.2 Meio-tom digital por difusão de erro

A maioria de impressoras jato de tinta ou laser pode imprimir somente minúsculos pontos pretos sobre o papel. Assim, qualquer imagem em níveis de cinzas deve ser primeiro convertido numa imagem binária antes que a impressão aconteça. Meio-tom (halftoning) simula níveis de cinza distribuindo apropriadamente pixels pretos e brancos. Técnicas de meio-tom populares são difusão de erro, excitação ordenada (ordered dithering) e máscaras de ruído azul [Knuth, 1987; Ulichney, 1987].

Nesta seção, exporemos brevemente a técnica de meio-tom por difusão de erro, pois essa técnica será utilizada em diversas partes deste curso.

Formalmente, dada uma imagem em níveis de cinza $G: \mathbb{Z}^2 \rightarrow [0,1]$ (isto é, com valores reais entre 0 e 1), o algoritmo de meio-tom deve gerar uma imagem binária $B: \mathbb{Z}^2 \rightarrow \{0,1\}$ tal que para todos os pixels (*i*, *j*) da imagem:

$$\overline{B}(i, j) \cong G(i, j),$$

onde $\overline{B}(i, j)$ é o valor médio da imagem *B* na vizinhança em torno do pixel (*i*, *j*).

Uma solução interessante para este problema foi introduzida por Floyd e Steinberg, que calcularam *B* a partir de *G* aplicando o seguinte algoritmo:

Para todos os pixels (*i*, *j*) da imagem *G* {
    Se (*G*(*i*, *j*) < 0,5) *B*(*i*, *j*) ← 0; senão *B*(*i*, *j*) ← 1;
    erro ← *G*(*i*, *j*) - *B*(*i*, *j*);
    G(*i*, *j*+1) ← *G*(*i*, *j*+1) + erro×alpha;
    G(*i*+1, *j*-1) ← *G*(*i*+1, *j*-1) + erro×beta;
    G(*i*+1, *j*) ← *G*(*i*+1, *j*) + erro×gamma;
    G(*i*+1, *j*+1) ← *G*(*i*+1, *j*+1) + erro×delta;
}

Isto é, o erro cometido ao aproximar *G*(*i*, *j*) pelo *B*(*i*, *j*) é propagado para os pixels vizinhos que ainda não foram processados. Desta forma, conserva-se o nível de cinza médio localmente. Normalmente, alpha + beta + gamma + delta vale um, para que o erro cometido seja inteiramente propagado pelos vizinhos. Evidentemente, vizinhanças mais amplas podem ser utilizadas. Os pesos de difusão de erro mais populares estão listados na tabela 1.

Tab. 1: Os pesos comumente utilizados no algoritmo de difusão de erro.

$\left[\frac{1}{16}\right] \times \begin{bmatrix} & \bullet & 7 \\ 3 & 5 & 1 \end{bmatrix}$	$\left[\frac{1}{48}\right] \times \begin{bmatrix} & & \bullet & 7 & 5 \\ 3 & 5 & 7 & 5 & 3 \\ 1 & 3 & 5 & 3 & 1 \end{bmatrix}$	$\left[\frac{1}{42}\right] \times \begin{bmatrix} & & \bullet & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{bmatrix}$
Floyd e Steinberg	Jarvis, Judice e Ninke	Stucki

3 Marcas d'Água de Autenticação para Imagens em Tonalidade Contínua

3.1 Câmera digital confiável de Friedman

Costuma-se citar o artigo [Friedman, 1993] como o trabalho que inspirou os primeiros trabalhos de marca d'água de autenticação. Na câmera digital proposta, deseja-se autenticar a imagem assim que esta emerge da câmera. Para isso, a câmera produz dois arquivos de saída para cada imagem capturada: a primeira é a própria imagem digital capturada pela câmera em algum formato; e a segunda é uma assinatura digital produzida aplicando a chave privada da câmera (que deve estar armazenada de forma segura dentro de algum microprocessador dentro da câmera). O usuário deve tomar cuidado para guardar os dois arquivos, para que possa autenticar a imagem mais tarde.

Uma vez que a imagem digital e a assinatura digital são geradas pela câmera e armazenadas no computador, a integridade e a autenticidade da imagem pode ser verificada usando um programa para decodificar a assinatura digital, que pode ser distribuído livremente aos usuários. O programa de verificação recebe como entrada a imagem digital, a assinatura digital e a chave pública da câmera, calcula hash da imagem digital, decriptografa a assinatura digital e verifica se as duas impressões digitais obtidas são iguais.

3.2 Localização da região de alteração

O esquema proposto por Friedman poderia ser melhorado de duas formas. A primeira seria embutir a assinatura digital no arquivo da imagem, o que eliminaria a necessidade de armazenar dois arquivos para cada imagem. Alguns formatos de imagem permitem armazenar alguns dados adicionais no cabeçalho ou rodapé do arquivo. Mas o mais interessante seria embutir a assinatura digital na própria imagem. A segunda seria permitir a localização da região alterada. Isto poderia ser interessante, por exemplo, para descobrir a intenção do falsificador ao adulterar a imagem.

3.3 Marca d'água de Yeung-Mintzer

Yeung e Mintzer propuseram em [Yeung, 1997] uma das primeiras técnicas de marca d'água de autenticação. Marca d'água é inserido pixel a pixel, de forma que a alteração pode ser localizada com precisão. Porém, como há apenas 1 bit de marca d'água para autenticar cada pixel, há 50% de chance da alteração de um único pixel passar despercebida. Porém, se uma região de tamanho razoável for alterada, muito dificilmente essa alteração passará despercebida. Este esquema funciona com chave secreta, isto é, a inserção e a detecção da marca d'água devem ser feitas utilizando a mesma chave. Demonstrou-se mais tarde que este esquema é completamente inseguro. Isto é, um falsificador poderia inserir a marca válida em qualquer imagem dispondo apenas de um pequeno conjunto de imagens validamente marcadas.

3.3.1 Inserção de marca d'água

Seja *B* uma imagem-logotipo binária a ser inserida na imagem-original *I* para produzir a imagem-marcada I'. A imagem-original *I* pode ser tanto em níveis de cinza (neste caso, vamos supor 8 bits ou 256 níveis de cinza por pixel) como colorida (neste caso, vamos supor 3 bytes por pixel no formato RGB). Vamos supor que a imagem-logotipo *B* seja do mesmo tamanho que a imagem-original *I*. Se os tamanhos das duas imagens forem diferentes, a imagem-logotipo *B* deve ser replicada ou redimensionada para que seja do mesmo tamanho que *I*.

Vamos descrever primeiro o caso da imagem em níveis de cinza. Tanto a inserção, quanto a extração da marca d'água depende de uma look-up-table (LUT) $k : \{0...255\} \rightarrow \{0,1\}$. Uma LUT *k* aleatória pode ser preenchida sorteando 256 valores booleanos. Esta LUT *k* funciona como uma chave secreta e deve ser mantida em segredo.

A inserção processa os pixels numa determinada ordem. Vamos supor que a ordem "raster" seja utilizada (isto é, processar os pixels linha por linha de cima para baixo e, dentro de uma determinada linha, da esquerda para direita).

Para inserir a marca d'água no primeiro pixel (1, 1), calcula-se $k(I(1,1))$. Se este valor for igual ao valor $B(1,1)$ da imagem-logotipo, não há nada que fazer. Se $k(I(1,1)) \neq B(1,1)$, o valor de $I(1,1)$ deve ser alterado para um nível de cinza próximo $I'(1,1)$ para obtermos $k(I'(1,1)) = B(1,1)$.

O erro cometido ao aproximar $I(1,1)$ para $I'(1,1)$ (isto é, $I(1,1) - I'(1,1)$) é espalhado para os pixels vizinhos, de forma semelhante ao bem conhecido algoritmo de difusão de erro utilizado para gerar imagens meio-tom. Isto assegura que o nível de cinza médio não é alterado localmente, o que garante uma alta qualidade visual à imagem marcada. Os autores usaram os pesos de difusão de erro abaixo, mas muitos outros valores poderiam ser utilizados:

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

Após a difusão de erro usando os pesos acima, obtemos novos valores de *I* na vizinhança do pixel (1, 1). Denotaremos a imagem obtida após o espalhamento de erro de $\bar{I}$. Assim, usando os pesos acima, obtemos os seguintes valores de $\bar{I}$ para a vizinhança de $I(1,1)$:

- $\bar{I}(2,1) = 0{,}5\big(I(1,1) - I'(1,1)\big) + I(2,1)$;
- $\bar{I}(1,2) = 0{,}5\big(I(1,1) - I'(1,1)\big) + I(1,2)$.

Agora, estamos prontos para processar o segundo pixel, digamos (1, 2), calculando a cor $I'(1, 2)$ semelhante a $\bar{I}(1, 2)$ de forma que $k(I'(1, 2)) = B(1, 2)$. O novo erro obtido é espalhado aos vizinhos. Este processo se repete até processar a imagem toda.

Para inserir marca d'água numa imagem colorida *I*, necessita-se de uma LUT pra cada plano de cor. Vamos denotá-las como k_R, k_G e k_B, respectivamente LUT do plano de cor vermelho, verde e azul. Para inserir a marca d'água num pixel (*i*, *j*), calcula-se a expressão booleana:

$$k_R(\bar{I}_R(i, j)) \otimes k_G(\bar{I}_G(i, j)) \otimes k_B(\bar{I}_B(i, j))$$

onde:

- $\otimes$ indica ou-exclusivo;
- $\bar{I}_R(i, j)$, $\bar{I}_G(i, j)$ e $\bar{I}_B(i, j)$ indicam o valor do pixel (*i*, *j*) da imagem $\bar{I}$ obtida difundindo o erro cometido ao aproximar *I* pela I', nos planos de cores vermelho, verde e azul, respectivamente.

Se o valor da expressão acima for igual a $b(i, j)$, nada a fazer. Se for diferente, os valores $\bar{I}_R(i, j)$, $\bar{I}_G(i, j)$ e/ou $\bar{I}_B(i, j)$ devem ser alterados para valores próximos $I'_R(i, j)$, $I'_G(i, j)$ e $I'_B(i, j)$ para que a expressão abaixo se torne igual a $b(i, j)$.

$$k_R(I'_R(i, j)) \otimes k_G(I'_G(i, j)) \otimes k_B(I'_B(i, j))$$

3.3.2 Extração de marca d'água

Dada uma imagem em níveis de cinza marcada I' e a LUT *k* utilizada na inserção de marca d'água, a imagem binária de checagem *C* pode ser extraída facilmente. Basta calcular:

$$C(i, j) \leftarrow k(I'(i, j))$$

para todos os pixels (*i*, *j*). Da mesma forma, dada uma imagem colorida marcada I', e três LUTs k_R, k_G e k_B, basta calcular:

$$C(i, j) \leftarrow k_R(I'_R(i, j)) \otimes k_G(I'_G(i, j)) \otimes k_B(I'_B(i, j)), \text{ para todos os pixels } (i, j).$$

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

3.3.3 Ataque de falsificação

Holliman e Memon [Holliman, 2000] apresentam o ataque de falsificação (counterfeiting attack) que pode subverter completamente a marca de Yeung-Mintzer. Isto é, tendo algumas poucas imagens marcadas utilizando uma LUT *k*, é possível marcar validamente uma imagem qualquer sem conhecer o valor de *k* ou a imagem-logotipo. Além disso, é possível calcular a chave secreta *k* a partir de algumas imagens marcadas com LUT *k*, conhecendo a imagem-logotipo *B*.

Forjar uma marca d'água de Yeung-Mintzer aproveita o fato de que cada pixel é autenticado independentemente de qualquer outro. Vamos expor o ataque somente para o caso níveis de cinza, porém a mesma idéia vale para o caso colorido.

Vamos supor que Mallory, um hacker malicioso, gostaria de inserir uma marca válida numa imagem *J* qualquer, sem conhecer a LUT *k* que é secreta. Vamos supor que Mallory de alguma forma conheça a imagem-logotipo *B* e tenha à disposição uma imagem I' onde a imagem *B* foi embutida utilizando a LUT *k*. O ataque torna-se mais fácil se Mallory dispuser de uma quantidade grande de imagens onde a imagem-logotipo *B* foi inserida utilizando a mesma LUT *k*. Porém, para simplificar a notação, assumiremos uma única imagem hospedeira I' (na verdade, se houver várias imagens hospedeiras, todas podem ser grudadas uma na outra para formar uma única imagem).

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

Se o tamanho da imagem I' for suficientemente grande para conter um pixel exemplar para cada nível de cinza (o que costuma acontecer na prática), a LUT secreta *k* pode ser completamente descoberta a partir dos subconjuntos S_0 e S_1. Basta associar a cada nível de cinza em S_0 o bit 0 e a cada nível de cinza S_1 o bit 1.

3.4 Marca d'água de Wong

Wong em [Wong, 1997] propôs uma outra marca d'água de autenticação, desta vez baseada em criptografia simétrica. Esse artigo foi melhorado em [Wong, 1998] para utilizar a criptografia de chave pública, tornando-se o primeiro trabalho de marca d'água de autentica-

ção de chave pública. O esquema de Wong consiste, basicamente, em dividir uma imagem em blocos e assinar cada bloco independentemente. Assim, é possível localizar o bloco onde a imagem foi alterada. Quanto menor o tamanho dos blocos, melhor a resolução de localização de alteração. A marca d'água é inserida nos bits menos significativos (LSB - least significant bit) da imagem. Assim, nas imagens em níveis de cinza, é possível inserir um bit em cada pixel. Para imagens coloridas, é possível inserir três bits em cada pixel. Para imagens com 256 níveis de cinza (8 bits por pixel), a alteração de LSB é visualmente imperceptível, pois equivale a somar ou subtrair um do nível de cinza.

Assim como o trabalho de Yeung-Mintzer, o trabalho de Wong possui defeitos sérios de segurança. Neste tutorial, estudaremos apenas a versão chave-pública de marca d'água de Wong. A versão chave-secreta é inteiramente análoga.

3.4.1 Inserção de marca d'água

A inserção de marca d'água numa imagem em níveis de cinza, usando esquema de Wong chave-pública, pode ser resumida como segue.

Passo 1: Seja I uma imagem em níveis de cinza a ser marcada, com $\boldsymbol{N} \times M$ pixels. Particione I em n blocos I_t $(0 \le t < n)$ de 8×8 pixels (no máximo, blocos na borda podem ser menores). Cada bloco I_t será marcado independentemente.

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

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

Passo 4: Calcule ou-exclusivo de H_t com B_t, obtendo a impressão digital marcada $\hat{H}_t$.

Passo 5: Criptografe $\hat{H}_t$ com a chave privada, gerando assim a assinatura digital S_t do bloco t.

Passo 6: Insira S_t nos LSBs de I_t^*, obtendo o bloco marcado I_t'.

3.4.2 Extração de marca d'água

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

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

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

Passo 3: Retire os LSBs de I'_t e decriptografe o resultado usando a chave pública, obtendo o bloco decriptografado D_t.

Passo 4: Calcule ou-exclusivo de H_t com D_t, obtendo o bloco de checagem C_t.

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

Aqui e no resto desta seção, o operador * indica limpar LSB e a marca ′ indica um bloco ou uma imagem com assinatura embutida.

Observe que, teoricamente, a imagem-logotipo *B* deveria estar disponível publicamente para efetuar a verificação de marca d'água. Na prática, porém, *B* é uma imagem com algum sentido visual (por exemplo, o logotipo da empresa) e qualquer alteração em I'_t irá muito provavelmente gerar um bloco de checagem C_t parecido com ruído aleatório, que não pode ser confundido com B_t mesmo que *B* não esteja disponível. A imagem *B* poderia ser até completamente preta (ou branca) e neste caso torna-se muito fácil disponibilizar *B* publicamente.

Li *et al.* [Li, 2000] sugerem uma ligeira variação do esquema acima. O seu método particiona cada bloco em duas metades. Depois, a metade à direita do bloco I_t^* é trocada com a metade à direita do próximo bloco $I^*_{(t+1) \bmod n}$ seguindo a ordem em zig-zag (figura 1) de forma que blocos vizinhos estão relacionados pelos dados fundidos. Cada bloco combinado é então assinado e inserido nos LSBs do bloco I_t^*. A mesma operação deve ser executada na verificação de marca d'água.

3.5 Ataques à marca d'água de Wong

Mostraremos a seguir algumas fraquezas criptoanalíticas dos métodos de Wong e Li e mostraremos os meios para torná-los robusto. Muitos resultados citados nesta seção foram obtidos pelo nosso grupo de pesquisa e estão relatados em [Barreto, 2002].

Em primeiro lugar, note que RSA de 64 bits, sugerido originalmente para uso com o esquema de Wong, é completamente inseguro. Uma RSA com chave de 64 bits pode ser fatorada em segundos usando um computador pessoal atual.

Um esquema de autenticação que consegue detectar quaisquer alterações na imagem marcada deve ser considerado mais seguro que um outro que não consegue detectar algumas formas de alterações, mesmo que estas alterações aparentemente não possam ser utilizadas para propósitos maliciosos. A mera existência de tais falhas indicam a fraqueza do esquema. Eles podem ser usados no futuro para atacar a marca d'água, mesmo que neste momento ninguém saiba como fazê-lo.

Por exemplo, Wong [Wong, 1998] sugeriu que generalizasse a sua técnica de marca d'água em níveis de cinzas para imagens coloridas simplesmente aplicando o método independentemente aos três planos de cores. Neste caso, a verificação da marca não irá detectar a troca dos planos de cores. Embora possa ser difícil de imaginar como este ataque poderia ser usado maliciosamente, é mais seguro que mesmo este tipo de alteração não passe despercebida. Este problema em concreto pode ser facilmente resolvido passando os três planos de cores em conjunto na função de hash.

3.5.1 Ataque recortar-e-colar e ataque de falsificação

Existe outro ataque muito simples, indetectável pelo esquema de Wong, que pode realmente ser utilizado com intenções maliciosas. Denominamos de ataque de recortar e colar. Suponha que Mallory, um hacker malicioso, possui uma coleção de imagens legitimamente marcadas, todas elas do mesmo tamanho e contendo a mesma imagem embutida *B* na marca d'água. Como cada bloco é marcado separadamente sem qualquer informação sobre a imagem hospedeira exceto suas dimensões, é possível que Mallory selecione blocos das imagens autênticas e construa com elas uma nova imagem cuja marca d'água será falsamente verificada como legítima. Aqui assumimos que as coordenadas originais de cada bloco são mantidas na imagem falsificada. Porém, em alguns casos (por exemplo, se o tamanho da imagem *B* for 4×4, 4×8, 8×4, 8×8, 8×16, etc.) pode até ser possível para recortar e colar dentro de uma imagem marcada mantendo a marca d'água inalterada. A figura 2 mostra um exemplo deste ataque.

Este ataque também se aplica para marca d'água de Li: o atacante deve somente copiar conteúdos sem LSB de dois semi-blocos de dois blocos vizinhos, digamos, I_t^* e I_{t+1}^*, e colá-los juntos com a assinatura digital que se encontra nos LSBs do bloco I_t'.

Se o ataque recortar-e-colar for aplicado repetidamente, uma imagem inteira falsificada mas com marca válida pode ser construída. Esta é exatamente a idéia de ataque de falsificação (counterfeiting) de Holliman-Memon. Vamos supor que Mallory deseja marcar uma imagem *J* tendo em mãos um banco de dado de imagens protegido pela marca de Wong. Mallory primeiro particiona *J* em blocos J_t. Vamos supor que B_t é a imagem-logotipo que

deve ser inserido no bloco J_t. Mallory procura, entre os blocos de banco de dados contendo marca d'água B_t, o bloco D_t' mais parecido a J_t. Então, insere o bloco D_t' no lugar de J_t. Repetindo este processo para todos os blocos de *J*, uma imagem falsificada (mas com marca d'água válida) pode ser construída. Este ataque pode ser aplicado com sucesso mesmo que usando um banco de dados relativamente pequeno. Holliman e Memon pegaram duas imagens de impressão digital em níveis de cinzas 750×750 da NIST, inseriram marca d'água de Wong num deles, e então construíram uma aproximação da segunda imagem convincente e corretamente marcada utilizando a primeira como banco de dados, isto é, utilizando somente 9000 blocos de banco de dados. Um ataque similar também pode ser efetuado contra marca d'água de Li. Mostraremos adiante que HBC1 torna impossíveis ataques de recortar-e-colar e de falsificação.

3.5.2 Ataque de aniversário simples

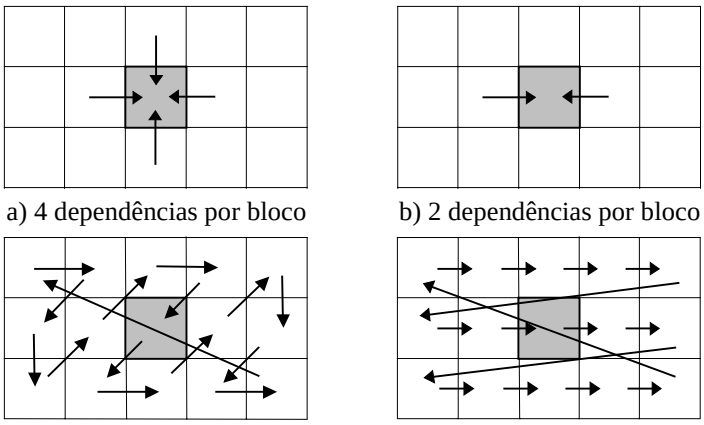
Ataque de aniversário [Menezes, 1997, seção 9.7] constitui um meio bem conhecido e poderoso para subverter assinaturas digitais. O atacante procura pelas colisões, isto é, pares de blocos que são levados a mesmo valor pela função de hash, portanto que têm a mesma assinatura. Usando função de hash que produz *m* valores possíveis, existe mais de 50% de chance de se achar uma colisão toda vez que aproximadamente $\sqrt{m}$ estiverem disponíveis. Esquema de Wong utiliza uma função de hash de não mais que 64 bits. Daí, espera-se que as colisões ocorram quando o atacante tiver coletado somente 2^{32} blocos. Em geral, a única proteção contra ataques de aniversário é aumentar o tamanho da função hash. Isto diminuiria a resolução de localização de alterações, pois os blocos devem ser maiores para hospedar mais dados inseridos. Mostraremos na próxima subseção que o ataque de aniversário clássico também se torna impossível sob HBC1.

Um cenário possível para ataque de aniversário é uma companhia de seguros que mantém um banco de dados de imagens de incidentes usando marca d'água de Wong para proteção de integridade e autenticidade das imagens. Um banco de dado típico de uma companhia de seguros grande pode conter mais de milhões de imagens com, digamos, 640×480 pixels, de forma que cada imagem é particionada em 4800 blocos individualmente assinados (de 8×8 pixels). Isto resulta em aproximadamente 2^{32} assinaturas, o suficiente para um ataque de aniversário.

Mallory, um hacker malicioso, deseja substituir um bloco assinado I_t' por um outro bloco *J* e prepara $r \approx 2^{32}$ variantes visualmente equivalentes $J_1,\ldots,J_r$ de *J*. Isto pode ser feito variando o segundo bit menos significativo de cada um dos 32 pixels arbitrariamente escolhidos de *J* (LSB não pode ser usada uma vez que a marca d'água será armazenada lá). Mallory então procura por um bloco de imagem D' no banco de dados que é levada ao mesmo valor que J_j pela função de hash, isto é

$$H(M, N, J_j^*) = H(M, N, D^*) .$$

A probabilidade de sucesso é maior que 0,5 por causa do paradoxo do aniversário. Este J_j (com a assinatura pega de D') pode substituir I_t' sem ser detectado pelo esquema de Wong. Se este processo é repetido um número suficiente de vezes, uma imagem inteira falsificada pode ser gerada. Ataque similar também pode ser executada contra watermarking de Li.



a) 4 dependências por bloco

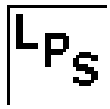
b) 2 dependências por bloco

c) 1 dependência por bloco (zig-zag)

d) 1 dependência por bloco (raster)

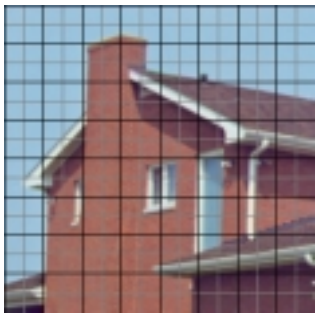
Fig. 1: O uso da informação contextual. Para calcular a assinatura de um bloco I_t (mostrado em cinza), o conteúdo de I_t e de seus blocos vizinhos são levados em conta. HBC utiliza 1 dependência por bloco, em ordem ou zig-zag ou raster.

a) Imagem original

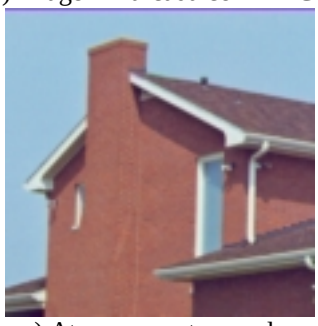


b) Imagem-logotipo 32×32

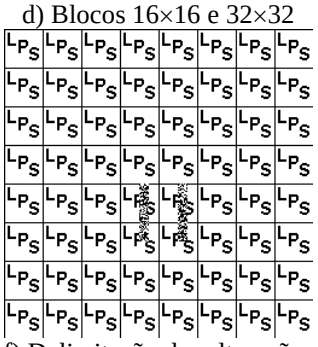
c) Imagem marcada com HBC2



d) Blocos 16×16 e 32×32



e) Ataque recortar e colar



f) Delimitação das alterações

Fig. 2: Impedindo ataque "recortar e colar" com HBC2. Uma imagem colorida 256×256 original (a) foi marcada usando chave privada e uma imagem logotipo 32×32 (b), gerando a imagem marcada (c). A imagem (d) mostra seus blocos constituintes. A imagem marcada (c) sofreu um ataque "recordar e colar" (e), indetectável pelo esquema de Wong. Usando HBC2, os blocos alterados podem ser localizados (f). Note que HBC2 detecta somente bordas dos blocos 16×16 alterados.

3.5.3 Hash block chaining versão 1

Conforme mostrado em [Barreto, 1999; Barreto, 2000; Holliman, 2000], a solução para impedir muitos ataques é introduzir uma informação contextual. Isto é, no cálculo da impressão digital H_t, alimentar a função de hashing *H* com os blocos vizinhos de I_t^*, além do próprio bloco I_t^* (veja a figura 1). Neste caso, se um bloco I_t' é alterado, a verificação da assinatura irá falhar em todos aqueles blocos que dependem de I_t', além do próprio bloco I_t'. Portanto, um número tão pequeno quanto possível de dependências é desejável para uma localização acurada de alterações na imagem. Idealmente, uma única dependência por bloco. O seguinte esquema implementa esta idéia:

$$H_t \equiv H(M, N, I_t^*, I_{(t-1) \bmod n}^*, t).$$

O índice do bloco *t* foi inserido para detectar rotação bloco a bloco. Assim como no esquema de Wong, os tamanhos *M* e *N* da imagem são inseridos para detectar cortes na imagem. Chamamos esta construção de *hash block chaining*, versão 1 (HBC1). Repetimos que um bloco I_t' for alterado, então HBC1 irá reportar que $I'_{(t+1) \bmod n}$ é inválido (além do próprio I_t').

Usando HBC1, o ataque recortar e colar simples não mais pode ser executado, pois se um bloco espúrio for colado no lugar de I_t', com probabilidade muito alta esta alteração irá introduzir uma alteração em $H_{(t+1) \bmod n}$. A probabilidade de que tal mudança não aconteça é de apenas $O(m^{-1})$. Esta alteração invalida a assinatura do bloco $I'_{(t+1) \bmod n}$. Assim, o ataque recortar e colar (e conseqüentemente, o ataque de falsificação) não pode ser mais executado.

De forma similar, se o ataque de aniversário for executado, o conteúdo alterado de I_t' induz com alta probabilidade uma mudança em $H_{(t+1) \bmod n}$. Assim, o atacante terá de forjar a assinatura de $I'_{(t+1) \bmod n}$ também, perpetrando um outro ataque. Mas este induz uma mudança em $I'_{(t+2) \bmod n}$. Portanto, o atacante irá defrontar com o problema da assinatura inválida propagar ciclicamente sobre todos os blocos, eventualmente destruindo a assinatura forjada do primeiro bloco falsificado.

3.5.4 Ataque de transplante

HBC1 é efetivo contra ataques recortar e colar, falsificação e aniversário. Mas não é seguro contra uma forma melhorada de ataque recortar e colar descrito abaixo. De fato, HBC1 ou qualquer outra técnica de partição que aumenta a função de hashing com contexto

determinístico e limitado dos blocos vizinhos são suscetíveis a o que nós chamamos um ataque de transplante. Para ver por que isto vale, sejam X' e $\overline{X}'$ duas imagens com marca d'água tipo HBC1. Vamos denotar o fato de que o hashing de um bloco X'_B depende do contexto do bloco X'_A (isto é, X^*_A) de $X'_A \to X'_B$. Suponha que as imagens X' e $\overline{X}'$ possuem blocos conforme mostrado abaixo:

$$\cdots \to X'_A \to X'_D \to X'_B \to X'_C \to \cdots,$$
$$\cdots \to \overline{X}'_A \to \overline{X}'_E \to \overline{X}'_B \to \overline{X}'_C \to \cdots,$$

onde $X^*_A = \overline{X}^*_A$, $X^*_B = \overline{X}^*_B$, $X^*_C = \overline{X}^*_C$ mas $X^*_D \neq \overline{X}^*_E$. Então, o par de blocos (X'_D, X'_B) pode ser trocado com o par $(\overline{X}'_E, \overline{X}'_B)$, sem ser detectado pelo esquema HBC1:

$$\cdots \to X'_A \to \overline{X}'_E \to \overline{X}'_B \to X'_C \to \cdots,$$
$$\cdots \to \overline{X}'_A \to X'_D \to X'_B \to \overline{X}'_C \to \cdots.$$

Imagens de documentos normalmente apresentam amplas áreas brancas, o que os torna muito suscetíveis a ataques de transplante. Por exemplo, se X'_A, X'_B, X'_C, $\overline{X}'_A$, $\overline{X}'_B$ e $\overline{X}'_C$ fossem todos blocos brancos sem ruído, o ataque teria sucesso facilmente. Note que simplesmente aumentar o número de dependências não previne o ataque de transplante. Se existirem duas dependências por bloco, como ilustrado abaixo, a tripla de blocos (X'_B, X'_E, X'_C) poderia ser trocada com a tripla $(\overline{X}'_B, \overline{X}'_F, \overline{X}'_C)$.

$$\cdots \leftrightarrow X'_A \leftrightarrow X'_B \leftrightarrow X'_E \leftrightarrow X'_C \leftrightarrow X'_D \leftrightarrow \cdots,$$
$$\cdots \leftrightarrow \overline{X}'_A \leftrightarrow \overline{X}'_B \leftrightarrow \overline{X}'_F \leftrightarrow \overline{X}'_C \leftrightarrow \overline{X}'_D \leftrightarrow \cdots.$$

Ataques semelhantes podem ser executados contra 4 dependências ou 8 dependências também.

3.5.5 Ataque de aniversário melhorado

HBC1 não pode resistir a um ataque de aniversário mais sofisticado. Este ataque substitui dois blocos consecutivos X'_t e X'_{t+1} pelos blocos forjados J_t e J_{t+1} (omitiremos "mod *n*" nos índices para simplificar a notação.) Três impressões digitais são afetadas por estas substituições: H_t (que depende de X'_t), H_{t+1} (que depende de ambos X'_t e X'_{t+1}), e H_{t+2} (que depende de X'_{t+1}). Suponha que o banco de dados tem *s* blocos assinados.

O atacante prepara *p* variantes visualmente equivalentes para J_t. Então, provavelmente $P \cong ps/m$ colisões para H_t serão encontradas (veja [Nishimura, 1990]). Mais explicitamente, *P*

pares (J_t^1, D_t^1), ..., (J_t^P, D_t^P) serão encontradas, onde J_t^1, ..., J_t^P são as variantes visualmente equivalentes de J_t e D_t^1, ..., D_t^P são blocos de banco de dados tais que a impressão digital de D_t^i é o mesmo que a impressão digital de J_t^i. Isto é:

$$H(M, N, D_t^{i*}, X_{t-1}^*, t) = H(M, N, J_t^{i*}, X_{t-1}^*, t), \text{ para } 1 \le i \le P.$$

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

De forma semelhante, o atacante prepara *q* variantes de J_{t+1}, provavelmente gerando $Q \cong qs/m$ colisões para H_{t+2}. Sejam (J_{t+1}^1, D_{t+2}^1), ..., (J_{t+1}^Q, D_{t+2}^Q) os pares que se colidem, isto é:

$$H(M, N, X_{t+2}^*, J_{t+1}^{j*}, t+2) = H(M, N, X_{t+2}^*, D_{t+2}^{j*}, t+2), \text{ para } 1 \le j \le Q.$$

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

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

$$H(M, N, J_{t+1}^{j*}, J_t^{i*}, t+1) = H(M, N, D_{t+1}^*, J_t^{i*}, t+1).$$

Então, se X_t e X_{t+1} são substituídas pelos blocos falsificados J_t^{i*} e J_{t+1}^{j*} e, ao mesmo tempo, as assinaturas dos blocos *t*, *t*+1 e *t*+2 são substituídos pelas assinaturas obtidas de LSBs de D_t^i, D_{t+1} e D_{t+2}^j, a adulteração passará despercebida por HBC1.

Qual deve ser os tamanhos *p* e *q* para que a chance de sucesso seja maior que 50%? Como existem pqs^2/m^2 pares de blocos e *s* blocos de banco de dados, uma colisão para H_{t+1} irá provavelmente ocorrer quando $(pqs^2/m^2)s \approx m$, isto é quando $pq \approx (m/s)^3$. Portanto, se o banco de dados possuir $s \approx \sqrt{m}$ assinaturas válidas, provavelmente dois blocos falsificados podem substituir dois blocos consecutivos válidos quando $p \approx q \approx m^{3/4}$ variantes visualmente equivalentes a cada bloco falsificado são preparadas.

3.5.6 Hash block chaining versão 2

Melhoramos HBC1 para opor aos ataques de transplante e de aniversário melhorado. Esta versão melhorada foi denominada HBC2 e faz uso de esquema de assinatura não-determinística. Alguns esquemas de assinaturas (por exemplo, DSA e Schnorr [Menezes, 1997, seção 11.5]) são não-determinísticos no sentido que cada assinatura individual depende não somente da função de hashing, mas também de algum parâmetro escolhido aleatoriamente. Usando um algoritmo de assinatura não-determinística, mesmo assinaturas de duas imagens idênticas serão diferentes. Esta propriedade efetivamente previne ataques de transplante. Uma assinatura determinística (como RSA) pode ser convertida numa não-determinística acrescentando "sal" (isto é, um dado arbitrário, estatisticamente único) à mensagem sendo assinada. HBC2 é definida como segue:

$$H_t \equiv H(M, N, I_t^*, I_{(t-1) \bmod n}^*, t, S_{t-1}),$$

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

O ataque de aniversário melhorado é completamente ineficaz contra HBC2, pois em HBC2 a assinatura de um bloco depende não somente do conteúdo do bloco vizinho, mas também da sua assinatura não-determinística. Vamos supor que um atacante tenha conseguido substituir dois blocos consecutivos válidos X_t e X_{t+1} por dois blocos falsificados J_t e J_{t+1}, e três assinaturas S_t, S_{t+1} e S_{t+2} por três assinaturas falsificadas (mas válidas) L_t, L_{t+1}, L_{t+2} enquanto mantém intacto o conteúdo do bloco X_{t+2}. Note que esta substituição é muito mais difícil em HBC2 do que em HBC1 devido à assinatura não-determinística e a dependência da assinatura. Mesmo neste cenário improvável, HBC2 irá reportar uma alteração, pois H_{t+3} depende não somente do conteúdo de $X_{t+2,}$ que não se altera, mas também da sua assinatura, que quase certamente muda.

O uso de HBC2 tem um surpreendente e agradável efeito colateral. Tipicamente, ataque de aniversário pode ser executado contra função hashing de comprimento m com um esforço de $O(\sqrt{m})$ passos. Porém, para HBC2 nenhum ataque que leva menos de $O(m)$ passos é conhecido. Portanto, parece que, num cenário otimista, o comprimento de hashing poderia ser cortado pela metade mantendo o nível de segurança original. Porém não recomendamos reduzir o comprimento do hashing até que esta conjectura seja analisada em maior profundidade, como tal redução poderia afetar a segurança do próprio algoritmo de assinatura.

HBC2 é capaz de detectar se algum bloco foi modificado, rearranjado, apagado, inserido, ou transplantado de uma imagem legitimamente assinada. Além disso, ou indica quais blocos foram alteradas ou, se uma grande região validamente marcada é copiada, onde ficam as bordas desta região. Chamamos atenção que a capacidade de localização é perdida se um

bloco (ou linha ou coluna) é inserido ou apagado, embora mesmo neste caso HBC2 irá reportar corretamente a presença de alguma alteração.

3.5.7 Discussões

Tipicamente, o comprimento de uma assinatura logaritmo discreto é aproximadamente duas vezes o tamanho da função hashing utilizada [Menezes, 1997, seção 11.5]. Isto é melhor que assinaturas RSA, cujo comprimento é sempre o da chave pública. Por exemplo, assinaturas DSA têm comprimento 320 bits, enquanto que as assinaturas RSA com nível de segurança equivalente devem ter aproximadamente 1024 bits. Neste sentido, assinaturas Schnorr são melhores para HBC2 [Menezes, 1997, seção 11.5.3], uma vez que eles conseguem redução máxima no tamanho da assinatura e portanto na quantidade de dados a serem incorporados na imagem hospedeira.

Experiências com HBC2 utilizando criptografia de curva elíptica resultaram tempos de assinatura e verificação de aproximadamente 10 segundos num Pentium-500, para imagens em níveis de cinza 512×512. A incerteza de localização de alteração foi menor que 0.2% da área da imagem.

3.5.8 Marca d'água de Wong-Memon

Wong e Memon propuseram em [Wong, 2001] um esquema de marca d'água muito semelhante a HBC2. O nosso trabalho reportado em [Barreto, 2002] foi desenvolvido independentemente do trabalho de Wong-Memon.

A diferença essencial entre os esquemas HBC2 e Wong-Memon é que o último utiliza um identificador $\mathtt{I}_I$ único para cada imagem I (por exemplo, um número seqüencial) que deve ser armazenado de alguma forma, fora da imagem. A existência desse identificador simplifica a construção de Wong-Memon, porém traz o desconforto ao usuário de ter que armazenar esse identificador de alguma forma (se é necessário armazenar este número serial, por que não armazenar a própria assinatura num arquivo independente?). A função hash de Wong-Memon torna-se:

$$H_t \equiv H(\mathtt{I}_I, M, N, I_t^*, t).$$

Note que desta forma não é necessário alimentar a função hash com informação de contexto.

4 Esteganografia para Imagens Binárias e Meio-tom

Uma vez que já estudamos algumas técnicas de autenticação de imagens em tonalidade contínua sem compactação, naturalmente aparece a curiosidade de querer estendê-las para as imagens binárias e para formatos de imagens compactadas. Neste tutorial, trataremos somente do primeiro caso.

Aparentemente, a autenticação de imagens binárias encontra-se ainda numa fase muito embrionária. Os estudos ainda estão limitados a como esconder os dados na imagem binária, sem chegar utilizar a informação escondida para autenticar a imagem binária. Conforme estudamos na seção anterior, para as imagens de tonalidade contínua havia uma forma "natural" de embutir os dados, que é inseri-los no bit menos significativo (LSB) de cada pixel. Ora, na imagem binária não existe o bit menos significativo do pixel, pois cada pixel é constituído por um único bit.

A esteganografia para imagens binárias pode ser subdividida em duas categorias: as técnicas para as imagens binárias "normais" e as técnicas para as imagens meio-tom.

Existem três formas básicas para embutir dados em imagens binárias: alterando valores de pixels individuais; mudando características um grupo de pixels; ou mudando características de um bloco de pixels.

A primeira abordagem troca pixel preto para branco ou vice-versa. A técnica DHST que será descrita adiante pertence a esta categoria.

A segunda abordagem modifica características tais como a posição do pixel superior esquerdo de cada componente conexo, a largura de pincelada, curvatura, etc. Esta abordagem normalmente depende do tipo de imagem e a quantidade de dados que pode ser inserida é limitada.

A terceira abordagem altera as características de um bloco da imagem. Por exemplo, poderia dividir uma imagem binária em blocos, digamos 8×8. Em cada bloco, um bit é embutido forçando o número de pixels pretos do bloco a ser par ou ímpar. Se o número de pixels pretos do bloco for par, convenciona-se que o bit zero está embutido naquele bloco. Se for ímpar, o bit um está embutido. Se um bloco já representar o bit que se deseja inserir, não há nada a fazer. Caso contrário, procura-se pelo pixel que causará a menor degradação visual segundo algum critério perceptual e troca-se o seu valor. Evidentemente, é possível estender a idéia para inserir dois ou mais bits por bloco. Uma outra técnica (mas que desta vez só se aplica para imagens meio-tom) orientada a blocos é alternar a matriz de pesos utilizada na difusão de erros de um bloco para outro [Pei, 2003; Hel-Or, 2001]. A imagem meio-tom é dividida em blocos e, dentro de cada bloco, utiliza-se uma determinada matriz de pesos (Floyd-Steinberg, Jarvis ou Stucki) para efetuar a difusão de erro. A matriz de pesos utilizado na geração da imagem meio-tom de um bloco pode ser determinada calculando a transformada de Fourier do bloco. Conforme a matriz utilizada, convenciona-se que está embutido bit zero ou um.

Nesta seção, apresentaremos mais detalhadamente duas técnicas para embutir informação em imagens meio-tom.

4.1 Imagem escondida torna-se visível com a sobreposição de duas imagens meio-tom

Uma classe de técnicas de esteganografia para imagens meio-tom consiste em esconder uma imagem binária em duas imagens meio-tom. As duas imagens meio-tom são visualmente "normais" quando vistas isoladamente. Porém se as duas imagens forem sobrepostas uma sobre outra e observadas contra a luz, é possível visualizar uma imagem binária escondida.

Aparentemente, as primeiras idéias utilizaram técnica de meio-tom denominada excitação ordenada (ordered dithering) e foram patenteadas [Knox, 1998; Wang, 1998].

Fu e Au apresentaram em [Fu, 2001] uma versão dessa idéia para imagens meio-tom obtidas pela difusão de erro. Esse método recebeu o nome de "data hiding by stochastic error diffusion - DHSED". Exporemos intuitivamente DHSED, sem entrar nos detalhes de implementação, pois já se conhece uma técnica melhor. A partir de uma imagem em níveis de cinza X, DHSED gera duas imagens meio-tom Y_1 e Y_2 de tal forma que, se Y_1 e Y_2 forem sobrepostas uma sobre a outra e vistas contra a luz, uma imagem binária escondida B se torna visível. A imagem Y_1 é gerada a partir da imagem X utilizando o algoritmo de difusão de erro usual. Para decidir a cor de um pixel $Y_2(i, j)$, analisa-se a qual das situações abaixo pertence o pixel (i, j):

1. $B(i, j)$ branco: $Y_2(i, j)$ recebe a mesma cor que $Y_1(i, j)$;
2. $B(i, j)$ preto e $Y_1(i, j)$ branco: Força para que $B_2(i, j)$ seja preto. Isto fará aparecer cor preta na posição (i, j) quando sobrepuser as imagens $Y_1(i, j)$ e $Y_2(i, j)$.
3. $B(i, j)$ preto e $Y_1(i, j)$ preto: A cor de $Y_2(i, j)$ pode ser tanto preto como branco. Portanto, deve-se atribuir a cor que minimiza o erro.

Olhando as regras acima, pode-se concluir que se a imagem em níveis de cinza X tiver tonalidade escura numa região branca da imagem B, a imagem B não poderá ser visualizada nitidamente sobrepondo as imagens Y_1 e Y_2. Assim, utilizando DHSED, não era possível visualizar nitidamente a imagem binária B escondida. Além disso, as duas imagens Y_1 e Y_2 deveriam ser obrigatoriamente imagens meio-tom correspondentes a uma única imagem em níveis de cinza X.

Num artigo muito recente (agosto de 2003), Pei e Guo [Pei, 2003] apresentam um melhoramento considerável para este tipo de esteganografia. A imagem B pode ser visualizada nitidamente tanto nas regiões escuras de X como nas regiões claras. Além disso, as imagens Y_1 e Y_2 podem ser inclusive imagens meio-tom de diferentes imagens em níveis de cinza.

O algoritmo de difusão de erro padrão está mostrado na figura 3. A variável $X(i, j)$ representa o pixel em níveis de cinza sendo processado, $v(i, j)$ é esse mesmo pixel após rece-

ber os erros difundidos pelos vizinhos, $Y(i, j)$ é a saída binária, e $e(i, j)$ é o erro cometido ao arredondar $v(i, j)$ para $Y(i, j)$. O erro $e(i, j)$ será espalhado para os vizinhos de acordo com a matriz dos pesos da difusão de erro h. A relação entre essas variáveis pode ser descrita como:

$$\begin{cases} v(i, j) = X(i, j) + X'(i, j), \text{ onde } X'(i, j) = \sum_{m=0}^{2} \sum_{n=-2}^{2} e(i+m, j+n) h(m, n) \\ e(i, j) = v(i, j) - Y(i, j) + N_B, \text{ onde } Y(i, j) = \begin{cases} 0, \text{ se } v(i, j) < 128 \\ 255, \text{ se } v(i, j) \geq 128 \end{cases} \end{cases} \quad (4.1)$$

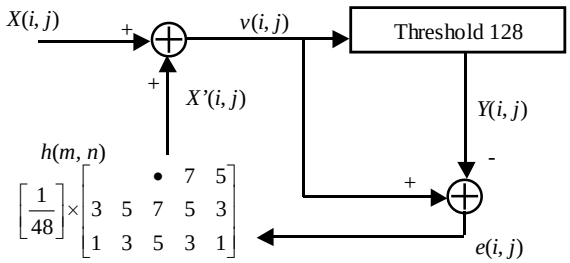


Fig. 3: Difusão de erro padrão.

A imagem meio-tom Y_1 é obtida processando a imagem em níveis de cinza X com o algoritmo de difusão de erro padrão, como na figura 3. Para gerar Y_2, deve-se verificar:

1. $B(i, j)$ preto e $Y_1(i, j)$ branco: $Y_2(i, j)$ é processado usando a seguinte equação em vez da equação 4.1:

$$\begin{cases} v(i, j) = X(i, j) + X'(i, j) - N_B \\ e(i, j) = v(i, j) - Y(i, j) + N_B \end{cases} \quad (4.2)$$

2. $B(i, j)$ preto e $Y_1(i, j)$ preto: utiliza-se a equação 4.1.
3. $B(i, j)$ branco e $Y_1(i, j)$ preto: aplica-se a equação 4.2.
4. $B(i, j)$ branco e $Y_1(i, j)$ branco: aplica-se a equação 4.2, só que invertendo os sinais de N_B.

O valor da variável N_B na equação 4.2 pode ser ajustado para controlar a qualidade de Y_2 e do padrão visual de B decodificado. Se o valor de N_B aumentar, a qualidade de Y_2 piora e o padrão visual decodificado irá tornar mais nítido; o oposto será verdadeiro quando o valor de N_B diminuir.

Pei e Guo apresentam uma série de generalizações ao esquema acima que descreveremos brevemente.

Primeiro, pode-se aumentar o número de imagens meio-tom para 3 ou mais. Evidentemente, a equação 4.2 deve ser alterada para lidar com mais de duas imagens meio-tom. A vantagem de se utilizar 3 ou mais imagens é que o padrão visual decodificado torna-se mais nítido à medida que se aumenta o número de imagens.

Segundo, Y_1 e Y_2 podem ser imagens meio-tom de imagens em níveis de cinza completamente diferentes. Fazendo isso, a qualidade do padrão visual decodificado deteriora. Assim, os autores sugerem uma técnica para enfatizar as bordas da imagem B de forma deixar mais nítida o padrão visual decodificado.

Terceiro, os autores propõe duas técnicas para autodecodificar a imagem B. O mais interessante delas consiste em pegar duas cópias de uma imagem meio-tom Y, deslocar uma delas de alguns pixels (por exemplo, 10 pixels horizontalmente), sobrepor as duas, olhar contra a luz e o padrão visual B irá aparecer. Este processo é possível pois o algoritmo de difusão de erro é um processo causal. Quando estiver processando um pixel (i, j), o valor de saída do pixel $(i, j\text{-}10)$ que se encontra 10 colunas atrás já estará determinado.

Esta técnica, quando estendida para imagens coloridas, fornece muitas outras possibilidades interessantes.

4.2 Inserção de dados modificando valores de pixels individuais

Uma outra forma de inserir informação numa imagem meio-tom é mudar os valores dos pixels individualmente. DHST (data hiding by self toggling) é uma dessas técnicas e é especialmente interessante pela sua simplicidade [Fu, 2000; Fu, 2002]. Essa técnica foi projetada originalmente para embutir dados em imagens meio-tom com pontos dispersos. A difusão de erro é uma das técnicas de meio-tom que gera imagens meio-tom com pontos dispersos.

Em DHST, um gerador de número pseudo-aleatório com uma semente conhecida é utilizado para gerar um conjunto de posições pseudo-aleatórias na imagem. Daí, um bit é embutido em cada posição forçando-o a ser ou preto ou branco. Com a probabilidade de 50%, o pixel da imagem meio-tom original é o valor desejado e não há necessidade de alteração. Com a probabilidade de 50%, o pixel tem valor contrário ao desejado e então o pixel deve ser alterado.

Para ler o dado escondido, simplesmente utiliza o mesmo gerador de número pseudo-aleatório com a mesma semente para obter as posições pseudo-aleatórias. Daí, basta ler os

valores dos pixels nessas posições. Evidentemente, DHST pode ser utilizado em qualquer imagem binária. Porém, neste caso, um ruído sal-e-pimenta visível irá aparecer.

Como DHST muda valores dos pixels individuais nas posições pseudo-aleatórias selecionadas, a intensidade local média pode ser afetada severamente. Para resolver este problema, Fu e Au [Fu, 2000] apresentam Data Hiding by Pair-Toggling (DHPT). A idéia desse algoritmo é na posição escolhida pseudo-aleatoriamente, a mudança de valor de um pixel é acompanhada, sempre que possível, pela mudança complementar de um vizinho. Por exemplo, se um pixel mestre é forçado mudar de 0 para 255, então os pixels vizinhos (na vizinhança 3×3) com valor 255 são identificados e um deles é escolhido aleatoriamente para mudar o seu valor para 0. Este pixel é chamado de pixel escravo.

No mesmo artigo, Fu e Au apresentaram Data Hiding by Smart Pair Toggling (DHSPT). Consiste basicamente em estabelecer uma regra para escolher o pixel escravo, entre os candidatos, de forma a perturbar o menos possível a qualidade visual da imagem meio-tom.

5 Conclusão

Neste tutorial, em primeiro lugar mostramos uma visão panorâmica das pesquisas atuais em marca d'água e esteganografia.

Em segundo, explicamos brevemente dois conceitos fundamentais para poder estudar de forma mais aprofundada uma série de marcas d'água: a assinatura digital e o meio-tom por difusão de erro.

Em terceiro lugar, estudamos detalhadamente as principais técnicas de marca d'água frágeis de autenticação para imagens em tonalidade contínua não-compactadas: Yeung-Mintzer, Wong, HBC2 e Wong-Memon. Estudamos uma série de ataques contra esses tipos de marca d'água: recortar-e-colar, aniversário, falsificação, aniversário melhorado, e transplante. Também estudamos os mecanismos para se proteger contra tais ataques.

Em quarto, argumentamos que embutir dados em imagens binárias possui uma dificuldade intrínseca. Demos uma visão panorâmica das técnicas para embutir dados em imagens binárias, entre elas duas técnicas orientadas a blocos. Estudamos detalhadamente duas outras técnicas para inserir dados em imagens meio-tom: difusão de erro estocástico e DHST (data hiding by self toggling).

Marca d'água e esteganografia é uma área de pesquisa extremamente ativa, e novas técnicas aparecem com grande freqüência. A finalidade deste tutorial foi dar um vislumbre de algumas das técnicas que estão sendo pesquisadas, sem ter a pretensão de esgotar o assunto.

6 Agradecimento

O autor gostaria de agradecer a FAPESP e a CNPq pelo apoio financeiro parcial deste trabalho através dos processos 2001/02400-9 e 300689/98-5, respectivamente.

Referências Bibliográficas

[Barreto, 1999] P. S. L. M. Barreto, and H. Y. Kim, "Pitfalls in Public Key Watermarking," *Sibgrapi - Brazilian Symp. Computer Graphics and Image Processing*, 1999, pp. 241-242.

[Barreto, 2000] P. S. L. M. Barreto, H. Y. Kim and V. Rijmen, "Um Modo de Operação de Funções de Hashing para Localizar Alterações em Dados Digitalmente Assinados," in Proc. Simpósio Brasileiro de Telecomunicações, paper 5150124, 2000.

[Barreto, 2002] P. S. L. M. Barreto, H. Y. Kim and V. Rijmen, "Toward a Secure Public-Key Blockwise Fragile Authentication Watermarking," *IEE Proc. Vision, Image and Signal Processing*, vol. 149, no. 2, pp. 57-62, 2002.

[Barreto, 2003] P. S. L. M. Barreto, *Criptografia Robusta e Marcas d'Água Frágeis: Construção e Análise de Algoritmos para Localizar Alterações em Imagens Digitais*, tese de doutorado, Escola Politécnica da Universidade de São Paulo, 2003.

[Friedman, 1993] G. L. Friedman, "The Trustworthy Digital Camera: Restoring Credibility to the Photographic Image," *IEEE T. Consumer Electronics*, vol. 39, pp. 905-910, Nov. 1993.

[Fu, 2000] M. S. Fu, and O. C. Au, "Data Hiding by Smart Pair Toggling for Halftone Images," *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, vol. 4, pp. 2318-2321, 2000.

[Fu, 2001] M. S. Fu, O. C. Au, "Data Hiding in Halftone Images by Stochastic Error Diffusion," *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, May 2001.

[Fu, 2002] M. S. Fu, and O. C. Au, "Data Hiding Watermarking for Halftone Images," *IEEE Trans. Image Processing*, vol. 11, no. 4, pp. 477- 484, 2002.

[Hel-Or, 2001] H. Z. Hel-Or, "Watermarking and Copyright Labeling of Printed Images," Journal of Electronic Imaging, col. 10, no. 3, pp. 794-803, 2001.

[Holliman, 2000] M. Holliman, and N. Memon "Counterfeiting Attacks on Oblivious Blockwise Independent Invisible Watermarking Schemes," *IEEE Trans. Image Processing*, 2000, vol. 9. no. 3, pp. 432-441.

[Knox, 1998] K. T. Know, "Digital Watermarking Using Stochastic Screen Patterns," United States Patent Number 5,734,752, 1998.

[Knuth, 1987] D. E. Knuth, "Digital Halftones by Dot Diffusion," *ACM Trans. Graph.*, vol. 6, no. 4, Oct. 1987.

[Li, 2000] C. T. Li, D. C. Lou, and T. H. Chen, "Image Authentication and Integrity Verification via Content-Based Watermarks and a Public Key Cryptosystem," *IEEE Int. Conf. Image Processing*, 2000, vol. 3, pp. 694-697.

[Menezes, 1997] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[Nishimura, 1990] K. Nishimura, and M. Sibuya, "Probability to Meet in the Middle," *J. Cryptology*, vol. 2, no. 1, pp. 13-22, 1990.

[Pei, 2003] S. C. Pei, and J. M. Guo, "Hybrid Pixel-Based Data Hiding and Block-Based Watermarking for Error-Diffused Halftone Images," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 867-884, 2003.

[Rivest, 1978] R. L. Rivest, A. Shamir, and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, pp. 120-126, 1978.

[Schneier, 1996] B. Schneier, *Applied Cryptography*, John Wiley & Sons, 1996.

[Ulichney, 1987] R. Ulichney, *Digital Halftoning*, The MIT Press, 1987.

[Wang, 1998] S. G. Wang, "Digital Watermarking Using Conjugate Halftone Screens," United States Patent Number 5,790,703, 1998.

[Wong, 1997] P. W. Wong, "A Watermark for Image Integrity and Ownership Verification," *IS&T PIC Conference*, (Portland, OR), May 1998 (also available as *Hewlett-Packard Labs. Tech. Rep.* HPL-97-72, May 1997).

[Wong, 1998] P. W. Wong, "A Public Key Watermark for Image Verification and Authentication," *IEEE Int. Conf. Image Processing*, 1998, vol. 1, pp. 455-459, (MA11.07).

[Wong, 2001] P. W. Wong, and N. Memon, "Secret and Public Key Image Watermarking Schemes for Image Authentication and Ownership Verification," *IEEE Trans. Image Processing*, vol. 10, no. 10, pp. 1593-1601, 2001.

[Yeung, 1997] M. M. Yeung, and F. Mintzer, "An Invisible Watermarking Technique for Image Verification," *IEEE Int. Conf. Image Processing*, 1997, vol. 1, pp. 680-683.