

# Capítulo 4:

## Marcas d'Água de Autenticação

### *Resumo e nossas contribuições*

Uma marca d'água é um sinal portador de informação embutido no dado digital que pode ser extraído mais tarde para fazer alguma asserção sobre o dado hospedeiro. 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 a integridade das imagens devem ser frágeis, para que qualquer alteração seja detectada. Este capítulo descreve as teorias que fundamentam as marcas d'água de autenticação e as nossas contribuições científicas nesta área.

Primeiro, descrevemos o conceito de assinatura digital, amplamente utilizada nas marcas de autenticação de chave pública. Em segundo lugar, descrevemos as principais marcas de autenticação para as imagens estáticas de tonalidade contínua (isto é, as imagens em níveis de cinza e coloridas): Yeung-Mintzer e Wong. Explicamos os principais ataques contra estas marcas e os meios para se defender contra eles. Em terceiro lugar, descrevemos as marcas de autenticação para as imagens binárias e meio-tom.

As nossas contribuições na área de marcas d'água de autenticação são:

- 1) *Hash block chaining*: Esta contribuição foi publicada em [Ri04; Ci04; Cn07; Cn05]. Nesta tese, ela está documentada na subseção 4.3.2. O principal responsável por esta contribuição foi o meu ex-orientando de doutorado Paulo S. L. M. Barreto, com a cooperação do Dr. Vincent Rijmen (da empresa Cryptomathic, Bélgica).

*Resumo*: As nossas pesquisas foram sobre as fraquezas criptográficas das marcas d'água para autenticação de imagem orientada a blocos. O algoritmo original de Wong [Wong, 1997; Wong, 1998], assim como vários outros algoritmos variantes, não são seguros contra um simples ataque recortar-e-colar ou o bem conhecido ataque de aniversário. Para torná-los seguros, foram propostos alguns esquemas para tornar a assinatura de cada bloco depender do conteúdo dos seus blocos vizinhos. Procuramos maximizar a resolução de localização das alterações, utilizando somente uma dependência por bloco através de um esquema que denominamos de “hash block chaining” versão 1 (HBC1). Mostramos que HBC1, assim como qualquer outro esquema dependente do contexto dos blocos vizinhos, é suscetível a uma outra técnica de falsificação que denominamos de ataque de transplante. Também mostramos um novo tipo de ataque de aniversário que consegue atacar HBC1. Para impedir esses ataques, propomos utilizar uma assinatura digital não-determinística junto com o esquema dependente de assinatura (HBC2). Finalmente, discutimos as vantagens de se utilizar assinaturas de logaritmo discreto em vez de RSA nas marcas de autenticação.

- 2) *Marcas de autenticação para imagens binárias*: Esta contribuição foi publicada em [Cn14] e está submetida em [Su02; Su04]. Nesta tese, elas estão documentadas nas subseções 4.4.2 e 4.4.3. Eu fui o principal responsável por esta contribuição, com a colaboração do meu orientando de mestrado Amir Afif e do prof. Ricardo de Queiroz da UnB.

*Resumo*: Na literatura, apenas um pequeno número de marcas de autenticação está disponível para as imagens binárias. Propomos duas novas marcas de autenticação para as imagens binárias: AWST e AWSF. A marca AWST é a-

propriada para as imagens meio-tom pontos dispersos e pode detectar qualquer alteração, mesmo uma alteração de um único pixel. A AWSF é apropriada para as imagens binárias em geral e pode detectar qualquer alteração visualmente significativa, ao mesmo tempo em que se mantém uma boa qualidade visual da imagem marcada. Esses algoritmos podem ser utilizados juntamente com a criptografia de chave secreta ou chave pública/privada. A segurança desses algoritmos baseia-se somente no segredo da chave. Na versão chave pública/privada, somente o dono da chave privada pode inserir a marca correta, enquanto que qualquer um pode verificar a autenticidade através da chave pública correspondente. Uma possível aplicação das técnicas propostas é na transmissão de fax pela internet, isto é, para a autenticação legal de documentos roteados fora da rede telefônica.

- 3) Temos outras contribuições em criptografia de chave pública [Ci08; Ci06; Ci03] que não estão documentadas nesta tese. Além disso, publicamos um curso tutorial sobre a marca de autenticação e a esteganografia [Rn01] cujo texto adaptado foi aproveitado nesta tese.

## 4.1 Introdução

O espetacular crescimento dos sistemas de multimídia interligados pela rede de computadores nos últimos anos (particularmente com o advento da World Wide Web) tem apresentado um enorme desafio nos aspectos tais como propriedade, integridade e autenticação dos 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 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 capítulo irá tratar somente das marcas para imagens digitais 2-D estáticas.

### ***Esteganografia***

O primeiro passo no estudo das marcas d'água é o estudo das técnicas utilizadas para embutir a informação numa imagem, conhecidas 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 se preocupar com a utilidade da informação escondida ou se a informação escondida é fácil ou difícil de ser removida. Algumas dificuldades especiais para inserir dados escondidos aparecem em tipos especiais de imagens, como nos formatos de imagens compactadas com perdas ou nas imagens binárias.

### ***Marcas robustas e frágeis***

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 a qual a marca será 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 robusta deveria ser possível de ser extraída mesmo que a imagem hospedeira sofra rotação, mudança de escala, mudança de brilho/contraste, compactação 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 robusta nela inserida deveria permanecer detectável mesmo após um processamento digital, enquanto a imagem resultante do processamento continuar visualmente reconhecível e identificável como a imagem original. Por esse motivo, as marcas d'água robustas são normalmente utilizadas para a verificação da propriedade (*copyright*) das imagens. Para dar um exemplo, se uma agência de notícias colocasse uma marca robusta numa fotografia, ne-

nhum 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 [Yeung and Mintzer, 1997; Wong, 1997; Wong, 1998; Wu and Liu, 1998; Li et al., 2000; Holliman and Memon, 2000]. Este tipo de marca d'água é útil para checar a integridade e a autenticidade da imagem, pois possibilita detectar alterações na imagem. 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 seja “marca d'água de autenticação”.

As marcas 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 detectado 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 semifrágil também serve para autenticar imagens. Só que estas procuram distinguir as alterações que modificam uma imagem substancialmente daquelas que não modificam o conteúdo visual da imagem. Uma marca semifrágil normalmente extrai algumas características da imagem que permanecem invariantes através das operações “permitidas” e as insere de volta na imagem de forma que a alteração de uma dessas características possa ser detectada.

### *Tipos de marcas de autenticação*

Podemos subdividir as marcas 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 as alterações não-intencionais na imagem tais como um erro de transmissão ou de armazenamento. Funciona como uma espécie de “check-sum”. Se o algoritmo de autenticação sem chave estiver disponível publicamente, qualquer pessoa pode inserir este tipo de marca em qualquer imagem e qualquer pessoa pode verificar se uma imagem contém uma marca válida.

A marca de autenticação com chave secreta (cifra simétrica) é usada para detectar uma alteração que pode ser inclusive 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 armazenado separadamente. Os algoritmos para inserção e detecção deste tipo de marca podem ser disponibilizados publicamente, e uma chave secreta é usada em ambas as 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  $k$ . Além disso, 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 a criptografia de chave pública para inserir uma assinatura digital na imagem. Usando uma cifra de chave pública, a autenticidade de uma imagem pode ser julgada sem a necessidade de se tornar pública qualquer informação privada.

***Marca de autenticação em imagens contones e binárias***

Existe uma forma “natural” de embutir as marcas de autenticação em imagens de tonalidade contínua (contone) não compactadas. É inserir os dados nos bits menos significativos (LSBs). Alterar os LSBs afeta muito pouco a qualidade da imagem, ao mesmo tempo em que se conhece exatamente os bits que serão afetados pela inserção da marca.

Não ocorre o mesmo com as imagens binárias. Numa imagem binária, cada pixel consiste de um único bit, de forma que não existe LSB. Isto traz dificuldades especiais para projetar marcas de autenticação para este tipo de imagem.

Inserir uma marca de autenticação em imagens contone compactadas com perdas também apresenta dificuldades especiais. Porém, este assunto não será tratado nesta tese.

***Exemplos de uso de marcas de autenticação de chave pública***

Entre os três tipos de marca de autenticação, a de chave pública é a que oferece mais recursos. Os possíveis usos de uma marca de autenticação de chave pública são enormes. Abaixo, citamos três exemplos:

- 1) *Câmera digital segura*. 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, 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 num circuito integrado dentro da câmera). O usuário deve tomar cuidado para guardar os dois arquivos, para que se 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. Ele calcula a função “hash” da imagem digital, decriptografa a assinatura digital e verifica se as duas “impressões digitais” obtidas são iguais. 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. A marca d'água de autenticação de chave pública pode ser usada para incorporar essas melhorias à câmera de Friedman.

- 2) *Autenticação de imagens distribuídas pela rede.* 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 da 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 será capaz de inserir a marca correta na imagem falsificada porque ele não conhece a chave privada da Alice. Além disso, Mallory não poderá distribuir uma foto sua como sendo da Alice porque ele não conseguirá assiná-la por desconhecer a chave privada da Alice.
- 3) *Fax confiável.* 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 trans-



missora, poderia verificar que o documento foi originado de uma máquina específica de FAX e que o documento não foi manipulado.

### ***Organização deste capítulo***

O restante deste capítulo está organizado como segue. A seção 4.2 apresenta o conceito de assinatura digital, amplamente utilizado nas marcas de autenticação. A seção 4.3 apresenta algumas marcas de autenticação para imagens contone, subdividida em duas subseções. A subseção 4.3.1 apresenta a marca de Yeung-Mintzer e a subseção 4.3.2 descreve a marca de Wong, os ataques contra esta marca e a nossa proposta para robustecer esta marca denominada “hash block chaining”. A seção 4.4 apresenta as marcas de autenticação para as imagens binárias e meio-tom, subdividida em 3 subseções. A subseção 4.4.1 é a introdução, a subseção 4.4.2 apresenta a marca de autenticação AWST (apropriada para as imagens meio-tom pontos dispersos) e a subseção 4.4.3 descreve marca de autenticação AWSF (apropriada para as imagens binárias em geral, exceto as imagens meio-tom pontos dispersos).

## 4.2 Assinatura Digital

Vamos apresentar nesta seção um conceito que é bastante utilizado nas marcas d'água de autenticação: a assinatura digital. Para isso, seguiremos de perto a redação didática de [Friedman, 1993] e [Barreto, 2003].

### *Criptografia simétrica*

A criptografia de chave secreta ou simétrica requer que tanto o transmissor quanto o receptor da 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.

### *Criptografia assimétrica*

O conceito de criptografia de chave pública foi inventado pelo W. Diffie e M. Hellman, e independentemente por R. Merkle [Schneier, 1996, chap. 19]. A criptografia de chave pública ou cifra assimétrica utiliza duas chaves: uma chave privada e outra 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 a sua chave pública conhecida publicamente. Qualquer pessoa que queira enviar uma mensagem secreta a Bob deve criptografar a mensagem usando esta chave pública e enviá-la a Bob. Bob, sendo único possuidor da chave privada, é a única pessoa capaz de decifrar a mensagem. Note que a necessidade de se combinar uma chave secreta entre o transmissor e o receptor foi eliminada.

### *Assinatura digital*

O processo descrito acima pode ser implementado “ao contrário”. Neste caso, a transmissora de mensagem Alice guarda uma chave privada, e a chave pública correspondente 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 as duas partes. Porém, fornece um meio para assegurar que as mensagens não foram forjadas: somente Alice, que possui a chave privada, poderia ter codificado uma mensagem que é decifrável pela correspondente chave pública.

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 as 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 a assinatura como o processo de verificá-la. O resultado é um dado (chamada assinatura digital e abreviada como DS) que acompanha a mensagem original. Desta forma, a mensagem original pode ser lida por todos, porém se um receptor chamado Bob desejar autenticá-la, Bob pode decriptografar a assinatura digital da mensagem usando a chave pública da Alice, recuperando a “impressão digital” da mensagem. Esta impressão digital deve ser idêntica à *hash* da mensagem original, se a mensagem não tiver sido adulterada.

### *Assinatura não-determinística*

Uma assinatura digital diz-se *determinística* se o seu valor for exclusiva e univocamente determinado pelos dados assinados e pela chave privada do signatário. Em contraste, uma assinatura digital é *não-determinística* se depender 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.

### **Algoritmo RSA**

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 et al., 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  $d \equiv e^{-1} \pmod{(p-1)(q-1)}$ , 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 as operações do RSA). Seja  $M \in \mathbb{Z}_n$  a mensagem a ser assinada. Uma assinatura RSA para  $M$  é definida como  $C = M^d \pmod{n}$ . A verificação da assinatura procede com a recuperação de  $M$  a partir de  $C$ :  $M = C^e \pmod{n}$ .

## 4.3 Marcas de Autenticação para Imagens Contone

### 4.3.1 Marca de Autenticação de Yeung-Mintzer

#### *Introdução*

Yeung e Mintzer [Yeung and Mintzer, 1997] propuseram uma das primeiras técnicas de marca d'água de autenticação. A marca é inserida pixel a pixel, de forma que a alteração pode ser localizada com precisão. Porém, como há apenas 1 bit de marca para autenticar cada pixel, há 50% de chance de uma alteração de um único pixel passar despercebida. Porém, se uma região de tamanho razoável for alterada, muito dificilmente essa alteração passará despercebida. Este esquema funciona com chave secreta, isto é, a inserção e a detecção da marca devem ser feitas utilizando a mesma chave. Demonstrou-se mais tarde 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.

#### *Inserção de marca de Yeung-Mintzer*

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 por pixel ou 256 níveis de cinza) 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 em níveis de cinza. Tanto a inserção, quanto a extração da marca depende de uma look-up-table (LUT)  $k: \{0..255\} \rightarrow \{0,1\}$ . Uma LUT  $k$  aleatória pode ser gerada 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 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 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  $I(1, 1)$ :

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

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 a marca d'água numa imagem colorida  $I$ , necessita-se de uma LUT para cada plano de cor. Vamos denotá-las como  $k_R$ ,  $k_G$  e  $k_B$ , respectivamente as LUTs dos planos de cores 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 os valores do pixel  $(i, j)$  da imagem  $\bar{I}$  obtida difundindo o erro, 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 os valores próximos  $I'_R(i, j)$ ,  $I'_G(i, j)$  e  $I'_B(i, j)$  para que a expressão abaixo se torne igual a  $b(i, j)$ :

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

### ***Extração da marca de Yeung-Mintzer***

Dada uma imagem em níveis de cinza  $I'$  marcada com a marca de Yeung-Mintzer e a LUT  $k$  utilizada na inserção da marca, 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.

### *Ataque de falsificação*

Holliman e Memon [Holliman and Memon, 2000] apresentaram 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 a tabela  $k$  ou a imagem-logotipo. Além disso, é possível calcular a chave secreta  $k$  a partir de algumas imagens marcadas com a tabela  $k$ , conhecendo a imagem-logotipo  $B$ .

Para forjar uma marca d'água de Yeung-Mintzer, aproveita-se do 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  $k$ . 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 disponível 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 a imagem  $J$ , Mallory divide os pixels de  $I'$  em dois subconjuntos disjuntos: o primeiro subconjunto  $S_0$  de pixels com valor zero na imagem-logotipo  $B$  e o segundo  $S_1$  de pixels com valor um em  $B$ . Como só existem 256 níveis de cinza, e uma imagem de tamanho usual possui centenas de milhares de pixels, provavelmente haverá exemplos de praticamente todos os níveis de cinza. Em cada pixel  $J(i, j)$ , deve ser embutido o bit  $B(i, j)$ . Para isso, Mallory procura, no subconjunto  $S_0$  ou  $S_1$  correspondente ao bit  $B(i, j)$ , o pixel com o nível de cinza mais próximo possível do  $J(i, j)$ . 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 ao utilizado no algoritmo de 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 em  $S_1$  o bit 1.

### 4.3.2 Marca de Wong e Hash Block Chaining

#### *Introdução*

Esta subseção descreve uma contribuição científica original nossa. O principal responsável pelas descobertas foi o meu ex-orientando de doutorado Paulo S. L. M. Barreto.

Wong [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 de autenticação 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 da alteração. A marca d'água é inserida nos bits menos significativos (LSBs - least significant bits) da imagem. Assim, nas imagens em níveis de cinza, é possível inserir um bit em cada pixel. Nas imagens coloridas, é possível inserir três bits em cada pixel. Para uma imagem com 256 níveis de cinza (8 bits por pixel), a alteração dos LSBs 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. Nesta subseção, estudaremos apenas a versão chave-pública da marca de Wong. A versão chave-secreta é inteiramente análoga.

***Inserção da marca de Wong***

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

*Passo 1:* Seja  $I$  uma imagem em níveis de cinza a ser marcada, com  $N \times M$  pixels.

Particione  $I$  em  $n$  blocos  $I_t$  ( $0 \leq t < n$ ) de  $8 \times 8$  pixels (no máximo, os blocos nas bordas 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 das bordas da imagem (cropping).

*Passo 4:* Calcule o 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  $I'_t$ .

***Extração da marca de Wong***

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

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

*Passo 2:* Seja  $I_t^*$  o bloco obtido de  $I_t'$  limpando os LSBs 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 o ou-exclusivo de  $H_t$  com  $D_t$ , obtendo o bloco de checagem  $C_t$ .

*Passo 5:* Se  $C_t$  e  $B_t$  (o 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 subseção, o operador  $*$  indica limpar os LSBs e a marca  $'$  indica um bloco ou uma imagem com a assinatura embutida.

Observe que, teoricamente, a imagem-logotipo  $B$  deveria estar disponível publicamente para efetuar a verificação da 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 et al., 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 4.1c) de forma que os 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 da marca d'água.

### ***Ataque recortar-e-colar e ataque de falsificação***

Mostraremos a seguir algumas fraquezas criptoanalíticas dos métodos de Wong e Li e mostraremos os meios para torná-los robustos.

Em primeiro lugar, note que a assinatura RSA de 64 bits, sugerida originalmente para ser usada com o esquema de Wong, é completamente insegura. 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 indica uma fraqueza do esquema. Elas podem ser usadas no futuro para atacar a marca d'água, mesmo que neste momento ninguém saiba como fazê-lo.

Por exemplo, Wong [Wong, 1998] sugeriu que a sua marca d'água em níveis de cinza fosse generalizada para as 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 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 alimentando os três planos de cores em conjunto na função de hash.

Existe um outro ataque muito simples, indetectável pelo esquema de Wong, que pode realmente ser utilizado com intenções maliciosas. Denominamos esse 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  $B$ . Como cada bloco é marcado separadamente sem qualquer informação sobre a imagem hospedeira exceto as suas dimensões, é possível que Mallory selecione alguns blocos das imagens autênticas e construa com eles uma nova imagem cuja marca d'água será falsamente verificada como legítima. 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  $4 \times 4$ ,  $4 \times 8$ ,  $8 \times 4$ ,  $8 \times 8$ ,  $8 \times 16$ , etc.) pode até ser possível recortar um bloco de uma imagem e colá-lo dentro da mesma imagem mantendo a marca d'água inalterada. A figura 4.2 mostra um exemplo deste ataque.

Este ataque também se aplica para a marca de Li: o atacante deve somente copiar os conteúdos sem os LSBs dos 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 uma marca válida pode ser construída. Esta é exatamente a idéia do 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 protegidas 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 do banco de dados contendo a marca  $B_t$ , o bloco  $D_t'$  visualmente mais parecido ao bloco  $J_t$ . 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 uma marca d'água válida) pode ser construída. Este ataque pode ser aplicado com sucesso mesmo usando um banco de dados relativamente pequeno. Holliman e Memon pegaram duas imagens de impressões digitais da NIST (750×750 pixels, em níveis de cinza), inseriram a marca de Wong num deles, e então construíram uma aproximação convincente da segunda imagem e corretamente marcada utilizando a primeira como o banco de dados, isto é, utilizando somente 9000 blocos validamente marcados como banco de dados. Um ataque similar também pode ser efetuado contra a marca de Li. Mostraremos adiante que HBC1 torna impossíveis os ataques recortar-e-colar e falsificação.

### *Ataque de aniversário simples*

O 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 uma de hash que produz  $m$  valores possíveis, existe mais de 50% de chance de se achar uma colisão toda vez que aproxi-

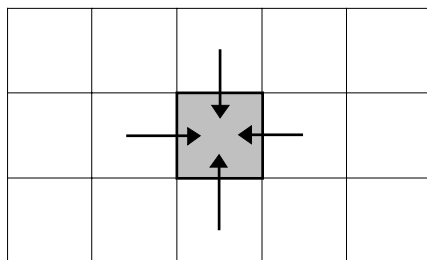
madamente  $\sqrt{m}$  blocos estiverem disponíveis. O esquema de Wong utiliza uma função de hash de não mais que 64 bits. Daí, espera-se que as colisões ocorram quando o atacante tiver coletado somente  $2^{32}$  blocos. Em geral, a única proteção contra o ataque de aniversário é aumentar o tamanho da função hash. Isto diminuiria a resolução de localização das alterações, pois os blocos devem ser maiores para hospedar mais dados inseridos. Mostraremos na próxima subsubseção que o ataque de aniversário clássico também se torna impossível sob HBC1.

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

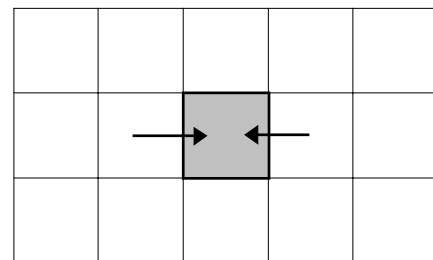
Mallory, um hacker malicioso, deseja substituir um bloco assinado  $I'_j$  por um outro bloco  $J$  e prepara  $r \approx 2^{32}$  variantes visualmente equivalentes  $J_1, \dots, J_r$  de  $J$ . Isto pode ser feito variando o segundo bit menos significativo de cada um dos 32 pixels arbitrariamente escolhidos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

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

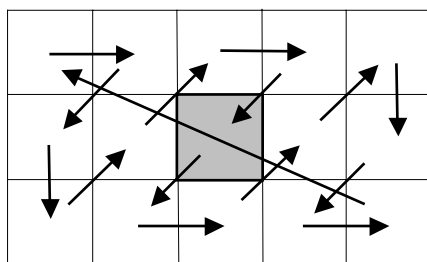
A probabilidade de sucesso é maior que 0,5 por causa do paradoxo do aniversário. O bloco  $J_j$  (com a assinatura pega de  $D'$ ) pode substituir o bloco  $I'_j$  sem ser detectado pelo esquema de Wong. Se este processo for repetido um número suficiente de vezes, uma imagem inteira falsificada pode ser gerada. Um ataque similar também pode ser executado contra a marca de Li.



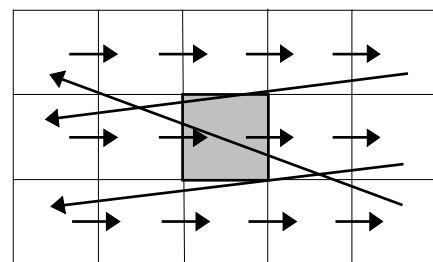
(4.1a) 4 dependências por bloco



(4.1b) 2 dependências por bloco



(4.1c) 1 dependência por bloco (zig-zag)

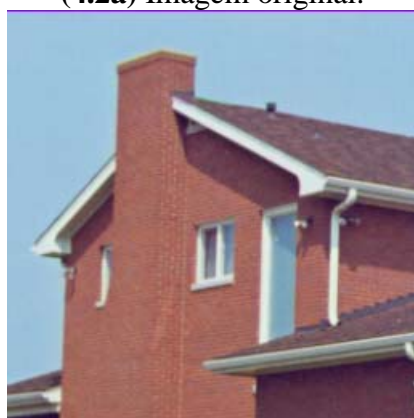


(4.1d) 1 dependência por bloco (raster)

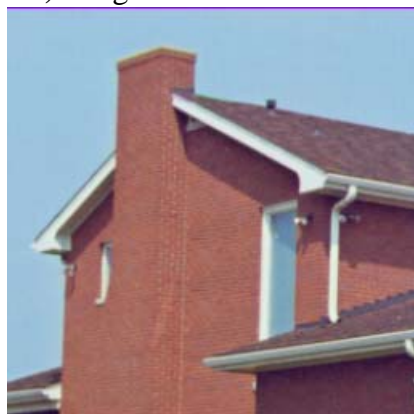
**Fig. 4.1:** Uso da informação contextual. Para calcular a assinatura de um bloco  $I_t$  (mostrado em cinza), o conteúdo do bloco  $I_t$  e de seus blocos vizinhos são levados em conta. O HBC utiliza 1 dependência por bloco, em ordem zig-zag ou raster.



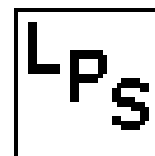
(4.2a) Imagem original.



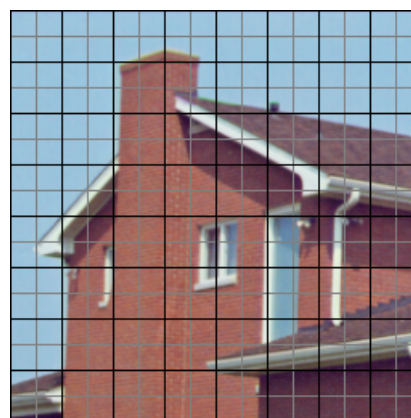
(4.2c) Imagem marcada com HBC2.



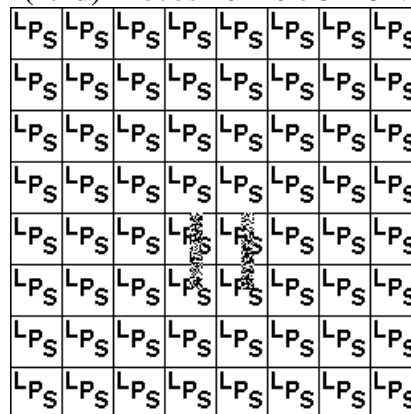
(4.2e) Ataque recortar-e-colar.



(4.2b) Imagem-logotipo 32×32.



(4.2d) Blocos 16×16 e 32×32.



(4.2f) Delimitação das alterações.

**Fig. 4.2:** Impedindo o ataque “recortar-e-colar” com HBC2. Uma imagem colorida 256×256 original (a) foi marcada usando a chave privada e uma imagem logotipo 32×32 (b), gerando a imagem marcada (c). A imagem (d) mostra os seus blocos constituintes. A imagem marcada (c) sofreu um ataque “recortar e colar” (e), indetectável pelo esquema de Wong. Usando o HBC2, os blocos alterados podem ser localizados (f). Note que o HBC2 detecta somente as bordas dos blocos 16×16 alterados.



***Hash block chaining versão 1***

Conforme mostrado em [Cn05; Cn07; Ri04; Holliman and Memon, 2000], a solução para impedir os ataques descritos anteriormente 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 4.1). Neste caso, se um bloco  $I'_t$  for alterado, a verificação da assinatura irá falhar em todos aqueles blocos que dependem de  $I'_t$ , além do próprio bloco  $I'_t$ . Portanto, um número tão pequeno quanto possível de dependências é desejável para uma localização acurada da alteração 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 a rotação bloco a bloco. Assim como no esquema de Wong, os tamanhos  $M$  e  $N$  da imagem são inseridos para detectar cortes na imagem. Chamamos esta construção de *hash block chaining*, versão 1 (HBC1). Repetimos que se um bloco  $I'_t$  for alterado, o HBC1 irá reportar que o bloco  $I'_{(t+1) \bmod n}$  é inválido (além do próprio  $I'_t$ ).

Usando o 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 acontecer 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 um 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 do bloco  $I'_{(t+1) \bmod n}$  também, perpetrando um outro ataque.

Mas isto induz uma mudança no bloco  $I'_{(t+2) \bmod n}$ . Portanto, o atacante irá defrontar com o problema de assinaturas inválidas propagarem ciclicamente sobre todos os blocos, eventualmente destruindo a assinatura forjada do primeiro bloco falsificado.

### *Ataque de transplante*

O HBC1 é efetivo contra ataques recortar-e-colar, falsificação e aniversário. Mas não é seguro contra uma forma melhorada do ataque recortar-e-colar descrita abaixo. De fato, o HBC1 ou qualquer outra técnica de partição que aumenta a função de hashing com contexto determinístico e limitado dos blocos vizinhos são suscetíveis ao que chamamos um ataque de transplante. Para isto, sejam  $X'$  e  $\bar{X}'$  duas imagens com marcas d'água tipo HBC1. Vamos denotar o fato da impressão digital de um bloco  $X'_B$  depender do conteúdo do bloco  $X'_A$  (isto é,  $X_A^*$ ) de  $X'_A \rightarrow X'_B$ . Suponha que as imagens  $X'$  e  $\bar{X}'$  possuam os blocos conforme mostrados abaixo:

$$\begin{aligned} \cdots \rightarrow X'_A \rightarrow X'_D \rightarrow X'_B \rightarrow X'_C \rightarrow \cdots, \\ \cdots \rightarrow \bar{X}'_A \rightarrow \bar{X}'_E \rightarrow \bar{X}'_B \rightarrow \bar{X}'_C \rightarrow \cdots, \end{aligned}$$

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

$$\begin{aligned} \cdots \rightarrow X'_A \rightarrow \bar{X}'_E \rightarrow \bar{X}'_B \rightarrow X'_C \rightarrow \cdots, \\ \cdots \rightarrow \bar{X}'_A \rightarrow X'_D \rightarrow X'_B \rightarrow \bar{X}'_C \rightarrow \cdots. \end{aligned}$$

As imagens de documentos normalmente apresentam amplas áreas brancas, o que as torna muito suscetíveis a ataques de transplante. Por exemplo, se  $X'_A$ ,  $X'_B$ ,  $X'_C$ ,  $\bar{X}'_A$ ,  $\bar{X}'_B$  e  $\bar{X}'_C$  fossem todos blocos brancos sem ruído, o ataque teria sucesso facilmente. Note que simplesmente aumentar o número de dependências não consegue evitar 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  $(\bar{X}'_B, \bar{X}'_F, \bar{X}'_C)$ .

$$\begin{aligned} \dots &\leftrightarrow X'_A \leftrightarrow X'_B \leftrightarrow X'_E \leftrightarrow X'_C \leftrightarrow X'_D \leftrightarrow \dots, \\ \dots &\leftrightarrow \bar{X}'_A \leftrightarrow \bar{X}'_B \leftrightarrow \bar{X}'_F \leftrightarrow \bar{X}'_C \leftrightarrow \bar{X}'_D \leftrightarrow \dots. \end{aligned}$$

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

### *Ataque de aniversário melhorado*

O esquema HBC1 também não consegue 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 tenha  $s$  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 and Sibuya, 1990]). Mais explicitamente,  $P$  pares  $(J_t^1, D_t^1), \dots, (J_t^P, D_t^P)$  serão encontrados, onde  $J_t^1, \dots, J_t^P$  são as variantes visualmente equivalentes de  $J_t$  e  $D_t^1, \dots, D_t^P$  são os blocos do 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:

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

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  $(J_{t+1}^1, D_{t+2}^1), \dots, (J_{t+1}^Q, D_{t+2}^Q)$  os pares que se colidem, isto é:

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

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 do banco de dados  $D_{t+1}$  tais 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  $X_{t+1}$  forem substituídos pelos blocos falsificados  $J_t^{i*}$  e  $J_{t+1}^{j*}$  e, ao mesmo tempo, as assinaturas dos blocos  $t$ ,  $t+1$  e  $t+2$  forem substituídos pelas assinaturas obtidas dos LSBs de  $D_t^i$ ,  $D_{t+1}$  e  $D_{t+2}^j$ , a adulteração passará despercebida pelo HBC1.

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

### **Hash block chaining versão 2**

Melhoramos o esquema HBC1 para resistir aos ataques de transplante e de aniversário melhorado. Esta versão melhorada foi denominada HBC2 e faz uso 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 uma assinatura não-determinística, mesmo as assinaturas de duas imagens idênticas serão diferentes. Esta propriedade efetivamente previne os 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. O esquema HBC2 é definido 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  $S_{-1} \equiv \emptyset$ . Note que não podemos usar  $S_{(t-1) \bmod n}$  porque quando a impressão digital  $H_0$  estiver sendo calculada, a assinatura  $S_{-1}$  ainda não será conhecida.

O ataque de aniversário melhorado é completamente ineficaz contra o HBC2, pois no HBC2 a assinatura de um bloco depende não somente do conteúdo do bloco vizinho, mas também da sua assinatura não-determinística. 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 no HBC2 do que no HBC1 devido à assinatura não-determinística e a dependência da assinatura. Mesmo neste cenário improvável, o 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 do HBC2 tem um surpreendente e agradável efeito colateral. Tipicamente, o ataque de aniversário pode ser executado contra uma função hashing de comprimento  $m$  com um esforço de  $O(\sqrt{m})$  passos. Porém, para o HBC2, nenhum ataque que leva menos de  $O(m)$  passos é conhecido. Portanto, parece que, num cenário otimista, o comprimento da função hashing poderia ser cortado pela metade mantendo o nível de segurança original. Porém não recomendamos reduzir o comprimento da hashing até

que esta conjectura seja analisada em maior profundidade, pois tal redução poderia afetar a segurança do próprio algoritmo de assinatura.

O HBC2 é capaz de detectar se algum bloco foi modificado, rearranjado, apagado, inserido, ou transplantado de uma imagem legitimamente assinada. Além disso, indica ou quais blocos foram alterados ou, se uma grande região validamente marcada for copiada, onde ficam as bordas da região alterada. Chamamos atenção que a capacidade de localização é perdida se um bloco (ou uma linha ou uma coluna) for inserido ou apagado, embora mesmo neste caso o HBC2 irá reportar corretamente a presença de alguma alteração.

### *Discussões*

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

As experiências com o HBC2 utilizando a criptografia de curva elíptica resultaram em tempos de assinatura e verificação de aproximadamente 10 segundos num Pentium-500, para as imagens em níveis de cinza 512×512. A incerteza de localização de alteração foi menor que 0,2% da área da imagem.

### *Marca d'água de Wong-Memon*

Wong e Memon [Wong and Memon, 2001] propuseram um esquema de marca d'água muito semelhante ao HBC2. O nosso trabalho reportado em [Ri04] 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  $\mathcal{S}_I$  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 esse número serial, por que não armazenar a própria assinatura num arquivo independente?). A função hash de Wong-Memon torna-se:

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

Note que desta forma não é necessário mais alimentar a função hash com a informação de contexto do bloco  $I_t^*$ .

## 4.4 Marcas de Autenticação para Imagens Binárias e Meio-Tom

### 4.4.1 Introdução

Uma vez estudadas 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 os formatos de imagens compactadas com perdas. Nesta tese, estudaremos somente o primeiro caso.

Nas seções anteriores, vimos que é praticamente impossível que uma marca de autenticação seja realmente segura sem estar apoiada na sólida teoria criptográfica. De fato, aquelas marcas d'água que não estavam fundadas em criptografia [Zhao and Koch, 1995; Yeung and Mintzer, 1997] or aquelas que aplicaram as técnicas criptográficas sem o devido cuidado [Wong, 1997; Wong, 1998; Li et al., 2000] tiveram mais tarde as suas fraquezas descobertas [Holliman and Memon, 2000; Ri04].

Numa marca de autenticação baseada em criptografia, o código de autenticação de mensagem (MAC) ou a assinatura digital (DS) de toda a imagem é computado e inserido na própria imagem. Porém, a inserção do código MAC/DS altera a imagem e conseqüentemente altera o próprio MAC/DS, invalidando a marca. Para evitar este problema, para as imagens em níveis de cinza ou coloridas, normalmente os bits menos significativos (LSBs) são apagados, calcula-se o MAC/DS da imagem com os LSBs apagados, e então o código é inserido nos LSBs. Em outras palavras, aqueles bits onde o código será inserido não são levados em conta ao se calcular o MAC/DS.

Para as imagens binárias ou meio-tom, esta idéia falha completamente, porque cada pixel possui um único bit. Modificando qualquer pixel para embutir a marca, a impressão digital da imagem é alterada, invalidando a marca. Conseqüentemente, embora haja muitos artigos sobre as técnicas para esconder dados em imagens binárias [Deseilligny and Le Men, 1998; Baharav and Shaked, 1998; Chen et al., 2000; Wu et



al., 2000; Fu and Au, 2000; Fu and Au, 2002a], conhecemos poucas marcas de autenticação baseadas em criptografia para as imagens binárias e meio-tom. Fu e Au [Fu and Au, 2002b] apresentam uma marca para detectar as alterações não-intencionais em imagens meio-tom, mas esta não pode ser considerada uma marca de autenticação porque não resiste a um ataque intencional ou malicioso.

De acordo com o paradigma criptográfico amplamente aceito, a segurança de uma marca de autenticação deve estar apoiada somente no segredo da chave. O fato de que uma imagem foi marcada, assim como o algoritmo utilizado para marcar a imagem devem poder se tornar públicos sem comprometer a segurança do esquema. Nas próximas subsubseções, propomos duas marcas de autenticação para as imagens binárias que satisfazem este requerimento, que denominamos de AWST (authentication watermarking by self toggling) e AWSF (authentication watermarking by shuffling and flipping). As marcas AWST e AWSF são apropriadas respectivamente para as imagens meio-tom pontos dispersos e as imagens binárias em geral. A AWSF não é adequada para as imagens meio-tom pontos dispersos, mas pode ser utilizada em imagens meio-tom pontos aglutinados. Assim, as duas técnicas podem ser usadas de forma complementar para proteger qualquer imagem binária.

As marcas AWST e AWSF podem ser usadas com criptografias de chave secreta ou pública. A AWSF de chave pública necessita de cuidados especiais para evitar um ataque que denominamos de “ataque de paridade”. Um uso possível de AWST/AWSF é em FAX seguro. Utilizando o FAX seguro, o receptor de um documento pode se certificar quem foi o gerador do documento, e que o documento não foi alterado (acidental ou maliciosamente) durante a transmissão.

Estas técnicas foram projetadas somente para autenticar as imagens digitais ortográficas. Para autenticar as imagens impressas, mais pesquisas são necessárias.

*Esteganografias para imagens binárias e meio-tom*

Existem três formas básicas de embutir dados em imagens binárias e meio-tom: alterar os valores dos pixels individuais, mudar as características de um grupo de pixels e mudar as características dos blocos da imagem.

A primeira abordagem troca as cores de determinados pixels [Fu and Au, 2000; Fu and Au, 2002; Tseng et al., 2002]. A técnica DHST, que será descrita adiante, pertence a esta categoria.

A segunda abordagem modifica as características tais como a posição do pixel superior esquerdo de cada componente conexo, a largura da pincelada, a curvatura, etc. [Maxemchuk and Low, 1997]. Esta abordagem normalmente depende do tipo de imagem e a quantidade de dados que pode ser inserida é limitada.

A terceira abordagem divide uma imagem em blocos e embute as informações através de alguma característica dos blocos da imagem. Por exemplo, poderia dividir uma imagem binária em blocos, digamos  $8 \times 8$ . Em cada bloco, um bit é embutido forçando o número de pixels brancos do bloco a ser par ou ímpar. Se o número de pixels brancos do bloco for par, convencionou-se que o bit zero está embutido naquele bloco [Wu et al., 2000]. Se for ímpar, o bit um está embutido. Se um bloco já representar o bit que se deseja inserir, não há nada a fazer. Caso contrário, procura-se pelo pixel que causará a menor degradação visual segundo algum critério perceptual e troca-se o seu valor. Evidentemente, é possível estender a idéia para inserir dois ou mais bits por bloco. Uma outra técnica orientada a blocos (mas que desta vez só se aplica para as imagens meio-tom) é alternar a matriz de pesos utilizada na difusão de erro de um bloco para outro [Pei and Guo, 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 utilizada na geração da imagem meio-tom de um bloco pode ser determinada calculando a transformada de Fourier do bloco. Conforme a matriz utilizada, convencionou-se que está embutido o bit zero ou um.

Para as imagens meio-tom, podemos citar ainda uma quarta abordagem: uma imagem é escondida em duas imagens meio-tom de forma que ela torna-se visível quando as duas são sobrepostas [Wang, 1998; Fu and Au, 2001; Pei and Guo, 2003].

#### **4.4.2 Marca de Autenticação AWST**

Esta subseção descreve uma contribuição científica original nossa. Eu fui o principal responsável pelas pesquisas descritas, contando com a colaboração do meu orientando de mestrado A. Afif.

##### *Técnica esteganográfica DHST*

DHST (data hiding by self toggling) é a técnica esteganográfica que se enquadra na primeira das quatro categorias listadas na subseção anterior [Fu and Au, 2000; Fu and Au, 2002a]. Ela é especialmente interessante pela sua simplicidade. Essa técnica foi projetada originariamente para embutir bits em imagens meio-tom pontos dispersos.

Na DHST, um gerador de números pseudo-aleatórios com uma semente conhecida é usado para gerar um conjunto de posições pseudo-aleatórias não repetidas dentro da imagem. Um bit é embutido em cada posição forçando-a a ser preta ou branca. Com a probabilidade de 50%, o pixel na imagem original tem o valor desejado e portanto nenhuma mudança é necessária. Com a probabilidade de 50%, o pixel tem o valor oposto ao desejado, e o pixel deve ser alterado. É importante garantir que não haja repetições de posições pseudo-aleatórias, pois neste caso o algoritmo tentaria inserir dois ou mais bits de informação num único pixel, o que evidentemente levaria a erro. Para ler o dado escondido, deve-se simplesmente gerar novamente as mesmas posições pseudo-aleatórias não-repetidas e ler os valores nessas localizações.

Evidentemente, a DHST pode também ser usada para qualquer imagem binária. Porém, neste caso, um ruído sal-e-pimenta se tornará visível. Neste artigo, iremos converter DHST numa marca de autenticação criptograficamente segura.

Como DHST muda os 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 and Au, 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 ser 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  $3 \times 3$ ) 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 algumas regras para escolher o pixel escravo, entre os candidatos, de forma a perturbar o menos possível a qualidade visual da imagem meio-tom.

### ***Marca de autenticação AWST***

Numa marca de autenticação segura utilizando alguma técnica para embutir dados em imagens binárias, deve-se calcular a função de hashing da imagem binária  $B$ , obtendo a impressão digital  $H = H(B)$ . A impressão digital  $H$ , depois de efetuar ou exclusivo e encriptar, torna-se a assinatura digital  $S$ . Esta assinatura digital deve ser inserida na própria imagem  $B$ , obtendo a imagem marcada  $B'$ . O problema é que, com a inserção da marca, a imagem  $B$  muda e conseqüentemente a sua impressão digital se altera. Isto é, teremos  $H(B) \neq H(B')$ . Como podemos superar esta dificuldade?

Apresentamos uma solução bem simples utilizando a DHST. Diferentemente da maioria de outras técnicas para embutir dados em imagens binárias, na DHST somente uns poucos bits são modificados e as posições desses bits são conhecidas tanto na fase de inserção como na fase de extração. Conseqüentemente, estes pixels podem ser zerados antes de calcular a função de hashing, da mesma forma que LSBs são zerados para marcar imagens em níveis de cinza. Vamos chamar a técnica assim obtida de AWST (authentication watermarking by self toggling). O algoritmo de inserção da marca AWST é:

1. Seja  $B$  a imagem binária a ser marcada e seja  $A$  a imagem logotipo binária a ser inserida em  $B$ .
2. Use um gerador de números pseudo-aleatórios com uma semente conhecida para gerar um conjunto de posições pseudo-aleatórias não-repetidas  $L$  dentro da imagem  $B$ .
3. Zere todos os pixels de  $B$  que pertencem a  $L$ , obtendo  $B^*$ .
4. Calcule a impressão digital  $H = H(B^*)$ .
5. Calcule o ou-exclusivo de  $H$  com  $A$ , obtendo a impressão digital marcada  $\hat{H}$ .
6. Criptografe  $\hat{H}$  com a chave secreta (criptografia simétrica) ou privada (criptografia assimétrica), gerando a assinatura digital  $S$ .
7. Insira  $S$  no conjunto de pixels  $L$ , gerando a imagem marcada  $B'$ .

O algoritmo de verificação da marca AWST é:

1. Seja  $X'$  a imagem marcada. Usando o mesmo gerador de números pseudo-aleatórios, gere novamente o mesmo conjunto de posições pseudo-aleatórias não-repetidas  $L$  onde a marca foi inserida.
2. Seja  $X^*$  a imagem obtida de  $X'$  zerando todos os pixels de  $L$ . Usando a mesma função de hashing, calcule a impressão digital  $H = H(X^*)$ .
3. Extraia a marca de  $X'$  lendo os pixels de  $L$  e decriptografando-os com a chave secreta (criptografia simétrica) ou pública (criptografia assimétrica), obtendo os dados decriptografados  $D$ .
4. Calcule o ou-exclusivo de  $H$  com  $D$ , obtendo a imagem de checagem  $C$ .
5. Se  $C$  e  $A$  são iguais, a marca está verificada. Caso contrário, a imagem marcada  $X'$  foi alterada.

A figura 4.3 ilustra o uso da marca d'água de autenticação AWST. Vamos supor que a imagem  $B$  (figura 4.3a) seja uma imagem suscetível a ataques a ser transmitida através de um canal pouco confiável, onde as alterações maliciosas podem ocorrer. Para proteger  $B$ , a imagem logotipo  $A$  (figura 4.3b) foi inserida em  $B$  usando o algoritmo AWST. A imagem  $B'$  (figura 4.3c) é a imagem marcada onde 1024 bits foram inseridos. Isto é suficiente para embutir uma assinatura digital RSA [Schneier, 1996].

Se executar o algoritmo de verificação, obtemos a imagem de checagem  $C$  (figura 4.3d) exatamente igual à imagem logotipo  $A$ . Mesmo que um único pixel de  $B'$  seja alterado, a imagem extraída será completamente ruidosa (figura 4.3f).

A figura 4.4 mostra a qualidade de um documento marcado com a AWST. Uma página de uma revista foi escaneada em 300 dpi, resultando numa imagem binária com 3318 linhas e 2536 colunas (figura 4.4a). As figuras 4.4b, 4.4c e 4.4d mostram respectivamente as imagens com 64, 320 e 1024 bits embutidos. Estas quantidades de bits são suficientes para inserir, respectivamente, um MAC com chave secreta, uma assinatura digital DSA e uma assinatura digital RSA.

### ***Resposta booleana***

Embora extrair uma imagem logotipo visível da imagem marcada possa ser fascinante, na realidade somente necessitamos receber uma resposta binária à seguinte pergunta: “a imagem marcada contém ou não uma marca válida?” Para obter esta resposta booleana, podemos eliminar o passo 5 do algoritmo de inserção da AWST e o passo 4 do algoritmo de verificação da AWST.

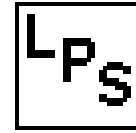
### ***Mantendo inalterado a intensidade média local***

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

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



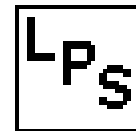
(4.3a) Imagem meio-tom  $B$  ( $512 \times 512$  pixels) a ser protegida com a marca de autenticação.



(4.3b) Imagem logo  $A$  ( $32 \times 32$  pixels) a ser inserida em  $B$ .



(4.3c) Imagem  $B'$  com marca d'água. 1024 bits foram inseridos.



(4.3d) Imagem logo extraída da imagem  $B'$ .

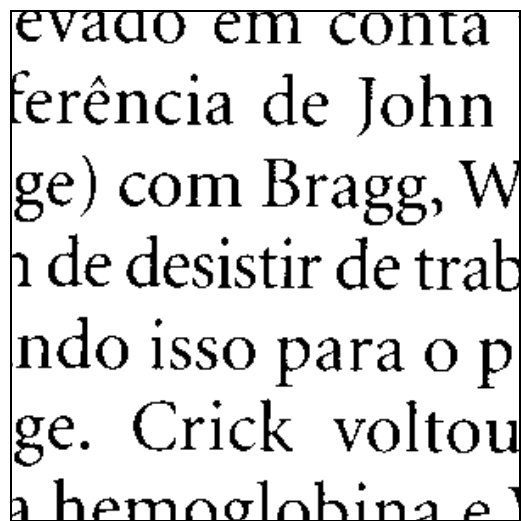


(4.3e) Imagem alterada  $X'$ .



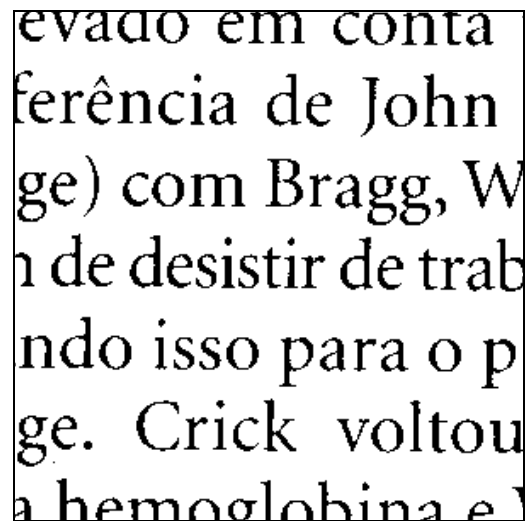
(4.3f) Imagem logo extraída de  $X'$ .

**Fig. 4.3:** Ilustração da AWST chave pública. A imagem logo  $A$  (b) foi inserida na imagem  $B$  (a) usando uma cifra de chave pública. A figura (c) mostra a imagem marcada. Executando o algoritmo de extração da marca, a figura (d) foi obtida. Se a imagem marcada for modificada mesmo que seja ligeiramente (e), uma imagem completamente aleatória é extraída (f).



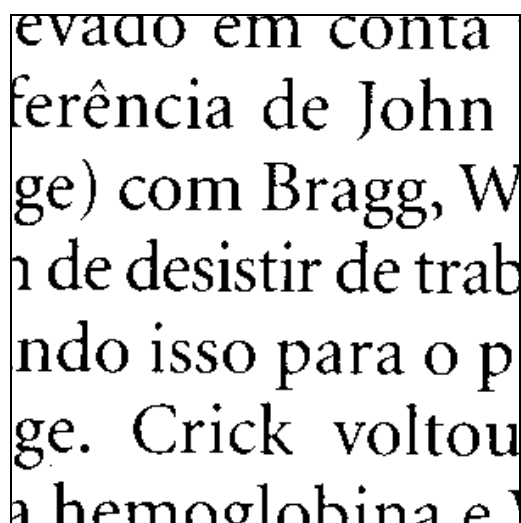
evado em conta  
ferência de John  
ge) com Bragg, W  
n de desistir de trab  
ndo isso para o p  
ge. Crick voltou  
a hemoglobina e

(4.4a) Parte da imagem original.



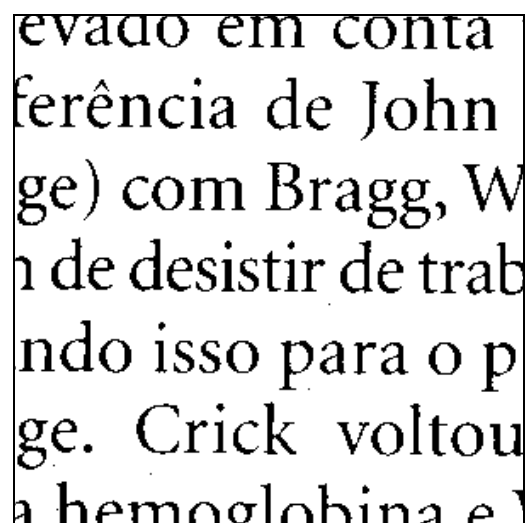
evado em conta  
ferência de John  
ge) com Bragg, W  
n de desistir de trab  
ndo isso para o p  
ge. Crick voltou  
a hemoglobina e

(4.4b) Imagem com 64 bits embutidos (apropriada para inserir um código de autenticação de mensagem de chave secreta).



evado em conta  
ferência de John  
ge) com Bragg, W  
n de desistir de trab  
ndo isso para o p  
ge. Crick voltou  
a hemoglobina e

(4.4c) Imagem com 320 bits embutidos (apropriada para inserir uma assinatura DSA).



evado em conta  
ferência de John  
ge) com Bragg, W  
n de desistir de trab  
ndo isso para o p  
ge. Crick voltou  
a hemoglobina e

(4.4d) Imagem com 1024 bits embutidos (apropriada para inserir uma assinatura RSA).

**Fig. 4.4:** Qualidade dos documentos marcados com AWST. (a) Uma página de uma revista foi escaneada em 300 dpi, resultando numa imagem binária com 3318 linhas e 2536 colunas. (b) A marca AWST, utilizando chave secreta, necessita inserir 64 bits na imagem. (c) Usando a assinatura DSA, 320 bits devem ser embutidos na imagem. (d) Usando a assinatura RSA, 1024 bits devem ser embutidos. Note que a degradação de imagem é baixa mesmo embutindo 1024 bits.



### 4.4.3 Marca de Autenticação AWSF

Esta subseção descreve uma contribuição científica original nossa. Eu fui o principal responsável pelas descobertas descritas, contando com a colaboração do prof. Ricardo de Queiroz da UnB.

#### *Técnica esteganográfica de Wu et al.*

Entre as técnicas de esteganografias para as imagens binárias, a de Wu et al. [Wu et al., 2000] é especialmente interessante. Ela pode ser aplicada a maioria das imagens binárias, pode embutir uma quantidade moderada de dados, e a qualidade visual de uma imagem marcada com esta técnica é excelente. Ela pode ser sumarizada como segue:

- 1) Divide a imagem  $Z$  a ser marcada em pequenos blocos (digamos,  $8 \times 8$ ).
- 2) A vizinhança de cada pixel (normalmente  $3 \times 3$ ) é analisada para calcular a “nota de impacto visual” (VIS - visual impact score). Por exemplo, os pixels na borda de um componente conexo terão VIS's baixas, enquanto que um pixel completamente cercado por pixels brancos (ou pretos) terá VIS alta.
- 3) Insira um bit em cada bloco, modificando (se necessário) o conteúdo do pixel dentro do bloco com a menor VIS, forçando o bloco a ter um número ímpar de pixels brancos (para inserir o bit 1) ou um número ímpar (para inserir o bit 0).
- 4) Como diferentes blocos podem ter diferentes quantidades de pixels com VIS's baixas (por exemplo, todos os pixels num bloco completamente branco ou preto terão VIS's altas), Wu et al. sugerem embaralhar a imagem  $Z$  antes de inserir os dados.

#### *AWSF versão 1*

O artigo [Wu et al., 2000] dedicou somente umas poucas linhas para afirmar que a técnica por eles proposta poderia ser usada para detectar alterações em documentos binários, sem dar detalhes técnicos.

A idéia óbvia de calcular o código MAC/DS da imagem toda e inseri-lo na mesma imagem falha porque a inserção irá modificar a impressão digital da imagem. A primeira idéia para inserir MAC/DS, sem modificar a impressão digital da imagem, é dividir a imagem em duas regiões: a primeira (pequena) região onde MAC/DS será inserido, e a segunda (grande) região onde a impressão digital será calculada. Vamos escrever esta idéia de forma algorítmica:

1) Seja dada uma imagem binária  $Z$ . Usando um gerador de números pseudo-aleatórios com uma semente fixa, construa uma estrutura de dados auxiliar chamada vetor de embaralhamento  $V$ , de forma que a imagem  $Z$  possa ser vista como uma seqüência de pixels  $\tilde{Z}$  completamente embaralhada. Na versão AWSF chave secreta, a própria chave secreta é usada como a semente do gerador pseudo-aleatório. Na versão chave pública/privada, deve-se tornar a semente conhecida publicamente. Vamos considerar um pequeno exemplo para deixar as idéias mais claras. Considere a seguinte imagem binária  $Z$  com somente  $3 \times 3$  pixels:

$$Z = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Seja o vetor de embaralhamento  $V$ , onde cada elemento é um índice (linha, coluna) para a imagem  $Z$ , dado por:

$$V = [(0,0);(1,2);(0,2);(2,1);(2,0);(0,1);(1,1);(1,0);(2,2)]$$

Então, a seqüência embaralhada de pixels  $\tilde{Z}$  (a imagem  $Z$  acessada através dos índices de  $V$ ) é:

$$\tilde{Z} = [1, 1, 1, 0, 1, 0, 1, 0, 0].$$

2) Seja  $n$  o comprimento do código MAC/DS adotado, e seja  $m$  o número de pixels de cada bloco. Divida a seqüência embaralhada  $\tilde{Z}$  em duas regiões:

- Primeira região  $\tilde{Z}_1$  com  $n \times m$  pixels, onde o MAC/DS será armazenado. Esta região está subdividida em  $n$  blocos com  $m$  pixels cada. Em cada bloco, um bit do MAC/DS será inserido.

- Segunda região  $\tilde{Z}_2$  com o restante da seqüência embaralhada  $\tilde{Z}$ . O algoritmo de inserção irá calcular a impressão digital desta região.
- 3) Usando uma função de hashing  $H$  segura do ponto de vista criptográfico, calcule a impressão digital da segunda região  $H = H(\tilde{Z}_2)$ . Criptografe a impressão digital  $H$  usando a chave secreta ou privada, obtendo o MAC/DS:  $S = K(H)$ .
  - 4) Insira  $S$  na primeira região, obtendo a imagem marcada  $Z'$ . Insira um bit de  $S$  em cada bloco, modificando (se necessário) o conteúdo do pixel do bloco com a menor VIS, para forçar o bloco a ter um número par/ímpar de pixels brancos.

O algoritmo de verificação AWSF1 aplicado a uma imagem marcada  $Z'$  é:

- 1) Calcule o mesmo vetor de embaralhamento  $V$  usado para a inserção. Note que na versão chave secreta, a chave é também a semente do gerador de números pseudo-aleatórios e conseqüentemente somente o proprietário da chave pode reconstruir o vetor de embaralhamento. Porém, na versão chave pública/privada, a semente é publicamente conhecida e conseqüentemente o vetor de embaralhamento também é publicamente conhecido.
- 2) Divida  $Z'$  em duas regiões  $Z'_1$  e  $Z'_2$ . Calcule a impressão digital  $H$  de  $Z'_2$ .
- 3) Extraia o MAC/DS armazenado em  $Z'_1$  e decriptografe-o usando a chave secreta ou pública, obtendo o dado de checagem  $D$ .
- 4) Se  $D = H$ , a marca está verificada. Caso contrário, a imagem  $Z'$  foi modificada ou uma chave incorreta foi usada.

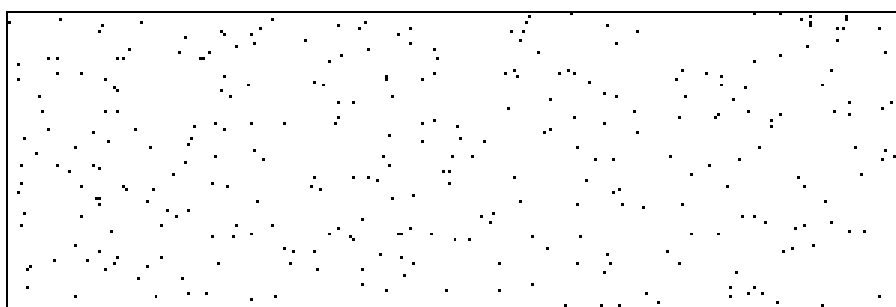
A figura 4.5a mostra parte de uma página da revista escaneada em 300 dpi, que poderia ser considerada como um documento binário “típico”. A figura 4.5b é a imagem correspondente depois de embutir 1024 bits usando a marca AWSF1, com blocos de 64 pixels.



(4.5a) Parte de uma página de uma revista escaneada em 300 dpi.



(4.5b) Parte da imagem com 1024 bits embutidos com AWSF.



(4.5c) Pixels pretos pertencem à região 1, e pixels brancos à região 2.

**Fig. 4.5:** Qualidade visual de um documento marcado com a marca AWSF.

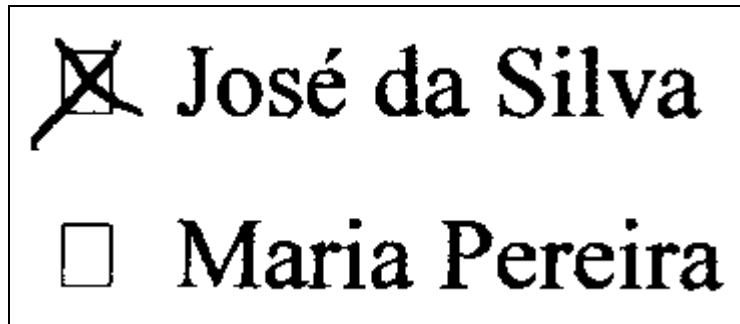
### *Ataque de paridade*

A marca AWSF1 consegue detectar qualquer alteração perpetrada na segunda região da imagem marcada, mesmo a modificação de um único pixel. De fato, a probabilidade de não detectar uma alteração nesta região é somente  $2^{-n}$  ( $n$  é o comprimento do MAC/DS adotado), o que pode ser desprezado.

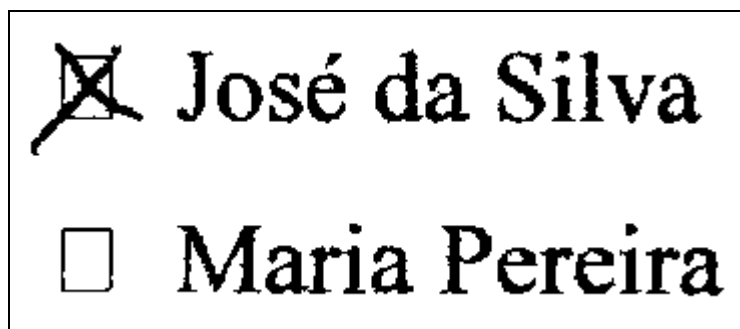
Infelizmente, uma alteração que mantenha a paridade dos blocos na primeira região não pode ser detectada pela AWSF1. Por exemplo, se dois pixels que pertencem ao mesmo bloco mudam os seus valores, a paridade deste bloco não será alterada e conseqüentemente esta alteração passará sem ser detectada. Denominamos este tipo de modificação de “ataque de paridade”.

Se a imagem marcada  $Z'$  for suficientemente grande, os pixels de  $Z'_1$  constituirão pixels isolados dispersos aleatoriamente na imagem  $Z'$  e é improvável que Mallory, um hacker malicioso, possa introduzir qualquer alteração visualmente significativa em  $Z'$  mudando somente os pixels de  $Z'_1$  (enquanto mantém a paridade de cada um dos blocos). Por exemplo, na figura 4.5c, os pixels pretos pertencem à região 1. Esses pixels estão inteiramente dispersos, e nenhuma alteração visualmente significativa poderá resultar modificando somente esses pixels.

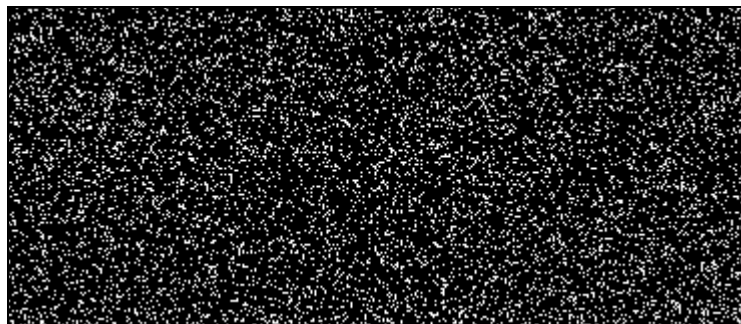
Entretanto, se a imagem  $Z'$  for pequena, os pixels de  $Z'_1$  podem formar regiões contíguas em  $Z'$ , o que levanta a possibilidade de que uma modificação visualmente significativa passe sem ser detectada pelo AWSF1. Por exemplo, a figura 4.6a é a imagem de uma cédula de votação e a figura 4.6b é a mesma imagem marcada com AWSF1. A figura 4.6c mostra pixels que pertencem à região 1 em preto. Qualquer pixel da região 1 pode ser modificado, desde que um outro pixel no mesmo bloco também seja modificado. Para obter uma imagem falsificada Mallory, um hacker malicioso, muda um pixel  $p$  do bloco  $i$ . Então, ele procura por um outro pixel no bloco  $i$  com a menor nota VIS e modifica o seu valor. A figura 4.6d mostra uma imagem construída repetindo esta idéia. Esta alteração não será detectada por AWSF1.



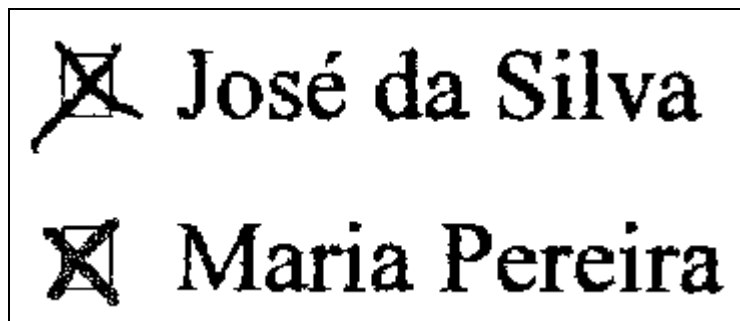
(4.6a) Uma pequena imagem (370×160) a ser marcada.



(4.6b) Imagem com 800 bits embutidos, usando blocos de 64 pixels.



(4.6c) Pixels pretos pertencem à região 1, onde um ataque de paridade pode ocorrer.



(4.6d) Imagem falsificada gerada pelo ataque de paridade, indetectável pela AWSF1.

**Fig. 4.6:** Falsificação “ataque de paridade”, indetectável pela marca AWSF1.

Na verdade, o cenário descrito acima somente se aplica à versão chave pública de AWSF1, onde as localizações das regiões 1 e 2, assim como a subdivisão da região 1 em blocos, são conhecidas publicamente.

Na versão chave secreta da AWSF1, não é necessário preocuparmos *muito* com o ataque de paridade, pois a chave secreta é usada para gerar o vetor de embaralhamento. Assim, Mallory não irá conhecer como a imagem marcada está dividida em regiões 1 e 2, e como a região 1 está subdividida em blocos. Entretanto, devemos nos preocupar *um pouco*, pois Mallory pode ter muitos meios diferentes pra obter pistas sobre as localizações das regiões e blocos. Por exemplo, vamos supor que Mallory tenha acesso a um banco de dados com muitos pares de documentos original e marcado com a AWSF1, todos com o mesmo tamanho e todos marcados usando a mesma chave secreta. Neste caso, ele terá conhecimento de que todos os pixels cujos valores são diferentes nos documentos original e marcado pertencem à região 1.

### ***AWSF versão 2***

Para minimizar a possibilidade de um ataque de paridade, sugerimos o seguinte melhoramento no passo 4 do algoritmo de inserção da AWSF1:

4) Insira  $S$  na primeira região utilizando o seguinte algoritmo, gerando a imagem marcada  $Z'$  :

Para  $i = 0$  até  $n-1$  {

Insira bit  $i$  de  $S$  no bloco  $i$ , forçando-o a ter um número par/ímpar de pixels brancos;

Calcule o novo MAC/DS  $S$ , alimentando a função de hashing com o conteúdo do bloco  $i$  e criptografando-o:

$S \leftarrow K(H(S, \text{pixels do bloco } i));$

}

Desta forma, o bloco  $n-1$  ainda pode sofrer um ataque de paridade. Porém, se o bloco  $n-2$  for modificado sem modificar a sua paridade, com 50% de chance esta modifica-

---

ção irá ser detectada. Se o bloco  $n-3$  for modificado (mantendo a paridade), há uma probabilidade de 75% de se detectar esta mudança. Se o bloco 0 for alterado (mantendo a sua paridade), existe uma probabilidade de  $1-2^{-(n-1)}$  de se detectar esta mudança. Assim, a AWSF2 certamente torna muito mais improvável que Mallory consiga perpetrar um ataque de paridade visualmente significativo com sucesso.



## 4.5 Conclusões

Neste capítulo, descrevemos as nossas contribuições científicas na área de marca d'água de autenticação.

Para isso, na seção “introdução”, definimos os conceitos necessários para a compreensão desta área.

Na seção “assinatura digital”, explicamos o funcionamento de uma assinatura digital, um conceito essencial no estudo das marcas de autenticação.

Na seção “marcas de autenticação para imagens contone”, descrevemos as marcas de autenticação de Yeung-Mintzer e de Wong, as fraquezas das ambas marcas, e a técnica hash block chaining, proposta por nós para robustecer a marca de Wong.

Na seção “marcas de autenticação para imagens binárias e meio-tom”, argumentamos que criar uma marca de autenticação para as imagens binárias possui dificuldades intrínsecas. Descrevemos as duas marcas de autenticação para imagens binárias propostas por nós para as imagens binárias e meio-tom denominadas, respectivamente, de AWST e AWSF.