Watermarking (marca d'água):

1) Data hiding ou esteganografia. -
- Blind: Consegue extrair a mensagem sem a imagem original.
- Não-blind: Precisa da imagem original para extrair a mensagem escondida.

1.1) Inserir mensagem no LSB (Least significant bit). Só funciona em imagens não-compactadas ou compactadas sem perdas.

O programa abaixo insere o arquivo texto *insere.cpp* dentro da imagem *lennag.tga* gravando o resultado como *marked.tga*.

Nota1: O programa não testa se *insere.cpp* cabe dentro de *lennag.tga*.

Nota2: O programa assume que no arquivo *insere.cpp* não aparece byte 0x00. Esse byte será usado para indicar o fim do arquivo.

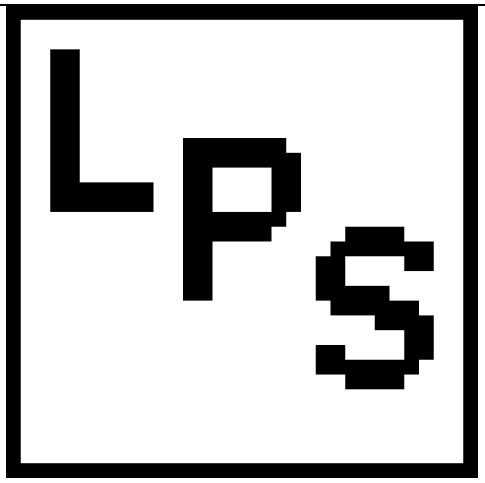
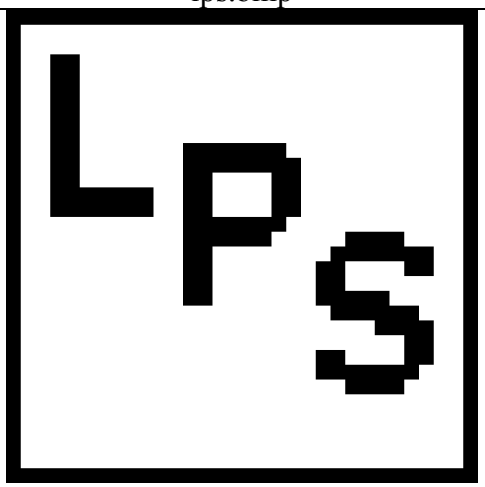
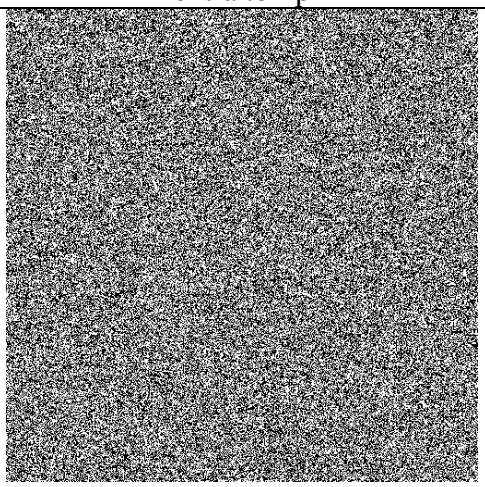
```
//insere.cpp pos-2011?
#include <proeikon>
int main()
{ IMGGRY a; le(a,"lennag.tga");
  for (int i=0; i<a.n(); i++)
    // a(i) = a(i) & 0xfe;
    a(i) = 2 * (a(i) / 2);
  // Aqui, todos os pixels de a sao pares.

  // Nota: Este programa nao verifica se o arquivo cabe na imagem
  FILE* arq=fopen("insere.cpp","r");
  int ch; int i=0;
  while ((ch=fgetc(arq))!=EOF) {
    for (int j=0; j<8; j++) {
      //a(i)=a(i) | (ch & 1);
      a(i) = a(i) + (ch % 2);
      ch = ch/2; i++;
    }
  }
  //O ultimo byte sera zero, pois LSB foi resetado
  imp(a,"marked.tga");
  fclose(arq);
}

//insere.cpp pos-2014
#include <cekeikon.h>
int main()
{ Mat_<GRY> a; le(a,"lennag.pgm");
  for (unsigned i=0; i<a.total(); i++)
    // a(i) = a(i) & 0xfe;
    a(i) = 2 * (a(i) / 2);
  // Aqui, todos os pixels de a sao pares.

  // Nota: Este programa nao verifica se o arquivo cabe na imagem
  FILE* arq=fopen("insere.cpp","r");
  int ch; int i=0;
  while ((ch=fgetc(arq))!=EOF) {
    for (int j=0; j<8; j++) {
      //a(i)=a(i) | (ch & 1);
      a(i) = a(i) + (ch % 2);
      ch = ch/2; i++;
    }
  }
  //O ultimo byte sera zero, pois LSB foi resetado
  imp(a,"marked.pgm");
  fclose(arq);
}
```

O programa abaixo extrai o arquivo-texto escondido na imagem *marked.tga* em *extrai.txt*.

```
//extrai.cpp pos-2011
#include <proeikon>
int main()
{ IMGGRY a; le(a,"marked.tga");
  FILE* arq=fopen("extrai.txt","w");

  int i=0;
  int ch;
  do {
    ch=0;
    for (int j=0; j<8; j++) {
      ch = ch/2;
      ch = ch + 128*(a(i)%2);
      //ch = ch >> 1;
      //ch = ch | ((a(i) & 1) << 7);
      i++;
    }
    if (ch!=0) fputc(ch,arq);
  } while (ch!=0);
  // Os bytes apos o arquivo serao todos zeros.
  fclose(arq);
}


//extrai.cpp pos-2014
#include <cekeikon.h>
int main()
{ Mat_<GRY> a; le(a,"marked.pgm");
  FILE* arq=fopen("extrai.txt","w");

  int i=0;
  int ch;
  do {
    ch=0;
    for (int j=0; j<8; j++) {
      ch = ch/2;
      ch = ch + 128*(a(i)%2);
      //ch = ch >> 1;
      //ch = ch | ((a(i) & 1) << 7);
      i++;
    }
    if (ch!=0) fputc(ch,arq);
  } while (ch!=0);
  // Os bytes apos o arquivo serao todos zeros.
  fclose(arq);
}
```

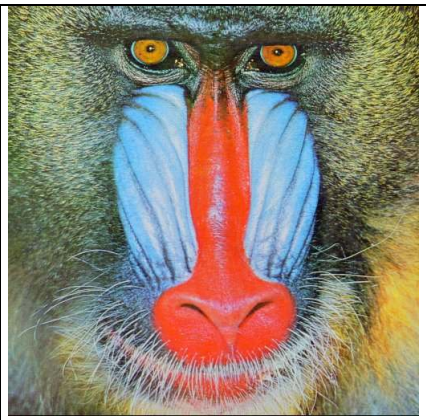
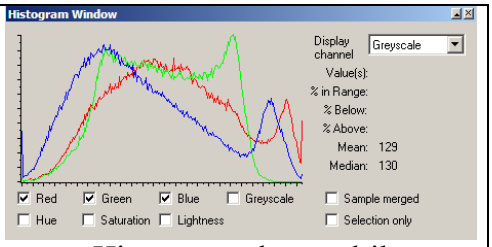
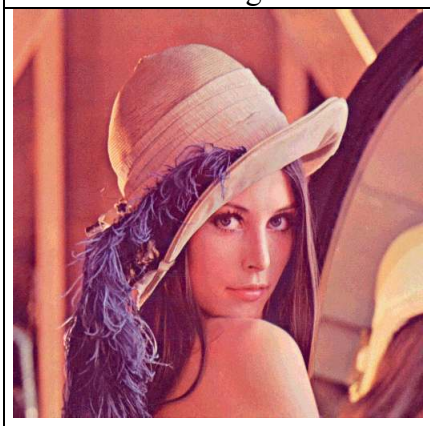
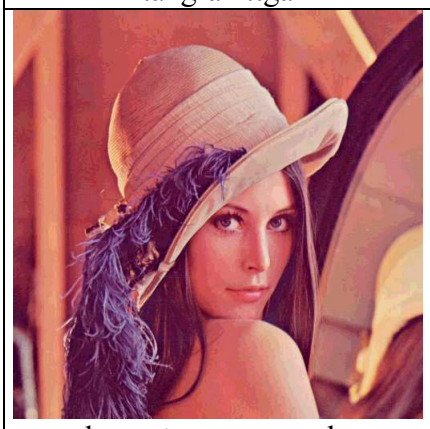
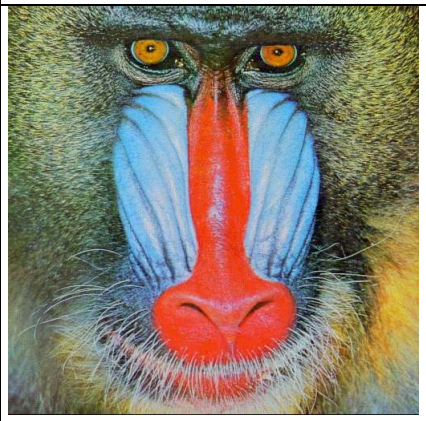
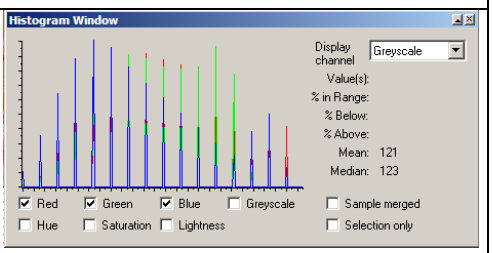
O programa abaixo insere uma imagem binária nos bits menos significativos da Lenna.

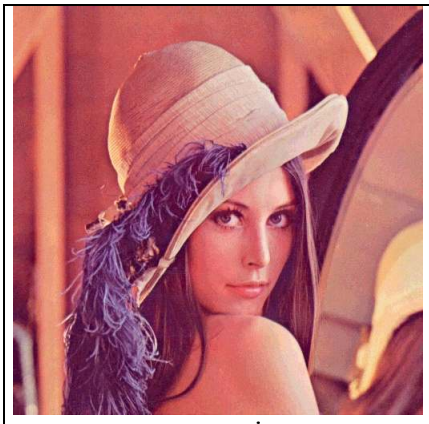
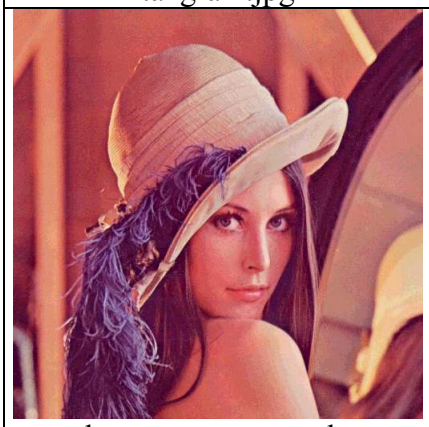
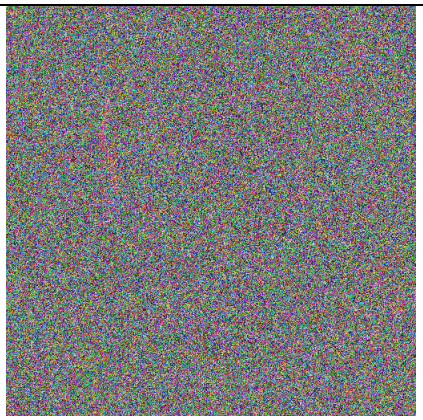
```
// marca.cpp
#include <proeikon>
int main()
{ IMGGRY a; le(a,"lennag.tga");
  IMGBIN b; le(b,"lps.bmp");

  IMGGRY c(a.nl(),a.nc());
  for (int i=0; i<c.n(); i++)
    c(i) = (a(i) & 0xfe) | b(i);
  imp(c,"marcado.tga");
}

//extrai.cpp
#include <proeikon>
int main()
{ IMGGRY d; le(d,"marcado.jpg");
  IMGBIN e(d.nl(),d.nc());
  for (int i=0; i<d.n(); i++)
    e(i)=d(i) & 1;
  imp(e,"extrai.bmp");
}
```

 lenna.tga	 lps.bmp
 marcado.tga	 extrai.bmp
 marcado.jpg A imagem marcado.tga foi compactado como marcado.jpg	 extrai.bmp A imagem extraída a partir de marcado.jpg.

1.2) Inserir uma imagem nos 4 bits menos significativos de uma outra imagem.

```
#include <proeikon>

BYTE embaralha(BYTE b)
{ int b8=(b >> 7) & 1;
  int b7=(b >> 6) & 1;
  int b6=(b >> 5) & 1;
  int b5=(b >> 4) & 1;
  int i = (b5 << 7) + (b6 << 6) + (b7 << 5) + (b8 << 4);
  return BYTE(i);
}

int main(int argc, char** argv)
{ if (argc!=5) {
    printf("Tangram: Esconde uma imagem colorida dentro de uma outra\n");
    printf("Tangram e ent-a.tga ent-b.tga tangram.tga\n");
    printf("  Esconde IMGCOR B dentro de IMGCOR A, gerando TANGRAM.TGA\n");
    printf("Tangram r tangram.tga sai-a.tga sai-b.tga\n");
    printf("  Recupera IMGCOR A e IMGCOR B escondidos em TANGRAM.TGA\n");
    erro("Erro: Numero de argumentos invalido");
  }

  char modo=tolower(argv[1][0]);
  if (modo!='e' && modo!='r') erro("Erro: Modo deve ser E ou R");

  if (modo=='e') {
    IMGCOR a; le(a,argv[2]);
    IMGCOR b; le(b,argv[3]);
    for (int l=0; l<a.nl(); l++)
      for (int c=0; c<a.nc(); c++) {

        a(l,c).r() = (a(l,c).r() & 0xf0) + (embaralha(b(l,c).r()) >> 4);
        // OU a(l,c).r() = 16*(a(l,c).r()/16) + (b(l,c).r()/16);
        a(l,c).g() = (a(l,c).g() & 0xf0) + (embaralha(b(l,c).g()) >> 4);
        a(l,c).b() = (a(l,c).b() & 0xf0) + (embaralha(b(l,c).b()) >> 4);
      }
    imp(a,argv[4]);
  } else {
    IMGCOR a; le(a,argv[2]);
    IMGCOR b(a.nl(),a.nc());
    for (int l=0; l<a.nl(); l++)
      for (int c=0; c<a.nc(); c++) {
        b(l,c).r() = embaralha(a(l,c).r() << 4);
        // OU b(l,c).r() = 16*(a(l,c).r()%16);
        b(l,c).g() = embaralha(a(l,c).g() << 4);
        b(l,c).b() = embaralha(a(l,c).b() << 4);
        a(l,c).r() = a(l,c).r() & 0xf0;
        // OU a(l,c).r() = 16*(a(l,c).r()/16);
        a(l,c).g() = a(l,c).g() & 0xf0;
        a(l,c).b() = a(l,c).b() & 0xf0;
      }
    imp(a,argv[3]);
    imp(b,argv[4]);
  }
}
```

 lenna.tga	 mandrill.tga	 Histograma de mandril
 tangram.tga	tangram.tga tem mandrill.tga escondido dentro.	
 lenna.tga recuperado	 mandrill.tga recuperado	 Histograma de mandril recuperado

 tangram.jpg	tangram.tga foi compactado com qualidade 95 e salvo como tangram.jpg	
 lenna.tga recuperado	 mandrill.tga recuperado	mandrill.tga recuperado a partir de lenna.jpg tem qualidade muito ruim.

2.1) SS (spread spectrum)

Para entender a idéia da marca d'água spread-spectrum, vamos fazer a seguinte consideração simples.

Suponha uma imagem A 512×512 inteiramente cinza (nível 128). Essa imagem pode ser imaginada como um vetor A com 262.144 elementos 128.

Considere um vetor P com 262.144 números [+1, -1, +1, -1, ...].

Seja:

$$Correlação(A,P) = \frac{1}{N}\sum_{i=0}^{N-1} A(i)P(i)$$

A correlação vai dar zero.

Considere a imagem $B(i) = 10P(i) + A(i)$, que corresponde à imagem A com marca d'água P com "força" 10. A imagem B será algo como [138, 118, 138, 118, ...]. Calculando correlação(B,P), isto é extraindo a marca d'água, obteremos 10.

Agora, vamos "introduzir ruído". Seja imagem A com 262.144 elementos cujos valores são aleatórios e uniformemente distribuídos no intervalo [0,255].

Seja o vetor P com 262.144 números aleatórios escolhidos independentemente do conjunto {-1, +1}, com probabilidades iguais.

A esperança da correlação(A,P) é zero.

Considere a imagem $B(i) = 10P(i) + A(i)$. A esperança da correlação(B,P) será 10.

A marca d'água inserida desta forma resiste a várias distorções:
1) Compactação com perdas tipo JPEG.
2) Ajuste de brilho/contraste.
3) Quantização.
4) Halftoning.
5) Pintar um pedaço da imagem.

Essa marca d'água não resiste a qualquer distorção que possa causar perda de "sincronismo":
1) Cropping de linhas ou colunas.
2) Rotação.
3) Ampliação ou redução.
4) Eliminação de linhas ou colunas.
5) Filtro passa-baixa ou mediana.

Provavelmente, colocando a marca d'água nas baixas freqüências, pode tornar resistente a filtro passa-baixa ou mediana.

```
C:\diretorio>img sss s b lennag.pgm marked.pgm 0.01 2014

C:\diretorio>img sss v b marked.pgm 2014 2013 2015 342
img sss v b marked.pgm 2014 2013 2015 342
       2014       0.00917201
       2013      -0.00138955
       2015      -0.00089227
        342      -0.00063218
```

A marca d´água não desaparece com:
1) Mudança de brilho/contraste
2) Riscos na imagem
3) Halftoning (com difusão de erro)
4) Compressão JPG
5) Inserção de outra marca d´água
6) Zerar bit menos significativo (fazer todos os pixels serem pares)

2.2) QIM (quantization index modulation)

1) Robusto:
1.1) Watermarking: - visível (logotipo da Rede Globo)
- invisível.

1.2) Fingerprinting:

Detecção de rotação/escala pela repetição espacial de watermark [Kutter].
Log-polar e log-log => rotação e escala tornam-se translações no domínio da transformada.

Há marca d'água robusta para imagens halftone que resiste à rotação/escala usando a idéia de [Kutter] (replicação espacial da marca).

Outros: halftone sobreposto.
mudança de matriz de difusão de erro.

2) Frágil:
2.1) Autenticação - supondo imagem descompactada ou compactada sem perdas
* [Wong] - autenticar imagens em níveis de cinza ou colorida descompactados ou compactados sem perda
[HBC]
[Celik] Hierárquico

imagem => índice de autenticação => criptografia => DS/MAC => data-hiding

2.2) Imagens binárias
- AWST: Authentication watermarking by self toggling. Escolhe pixels pseudo-aleariamente e muda o valor deles. Se a imagem não for halftone, aparece ruído sal e pimenta.
- AWTR: Authentication watermarking by template ranking.
- AWTC: Authentication watermarking by template ranking with symmetrical central pixels.
- Inserir vários bits alterando um único bit.

2.3) Compactada com perdas
- colorida e grayscale: JPEG, JPEG2000
- vídeo: MPEG, WMV, etc.
- binárias: JBIG2.

2.4) Semi-frágil
hashing robusto ou perceptual

2.5) Marca reversível

Referências:

Artigos sobre watermarking spread-spectrum:

[Cox1997] I. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia," IEEE T. Image Processing, vol. 6, no. 12, pp. 1673-1687, 1997.

Artigo sobre QIM (quantization index modulation):

[Chen2001] B. Chen, G. W. Wornell, "Quantization index modulation: a class of provably good methods for digital watermarking and information embedding," IEEE T. Information Theory, Vol. 47, No. 4, pp. 1423-1443, May 2001.

Marca d'água que resiste à rotação e escala:

[Kutter1998] M. Kutter, "Watermarking resisting to translation, rotation and scaling," Proceedings of SPIE , November 1998.

Artigos sobre watermarking de autenticação para imagens em níveis de cinza:

[Ri04] 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.

[Rn01] H. Y. Kim, “Marcas d’Água Frágeis de Autenticação para Imagens em Tonalidade Contínua e Esteganografia para Imagens Binárias e Meiotom,” Revista de Informática Teórica e Aplicada, Instituto de Informática da UFRGS, vol. 10, no. 1, pp. 97-125, 2003.

[T05] tes4.pdf (Marcas d'Água de Autenticação)

Artigos sobre watermarking de autenticação para imagens binárias:

[Pi01] H. Y. Kim, S. V. D. Pamboukian, P. S. L. M. Barreto, “Authentication Watermarkings for Binary Images,” in Chang-Tsun Li (ed.), Multimedia Forensics and Security, IGI Global, pp. 1-23, 2008.

[Ri06] H. Y. Kim, and A. Afif, “Secure Authentication Watermarking for Halftone and Binary Images,” *Int. J. Imaging Systems and Technology*, vol. 14, no. 4, pp. 147-152, 2004.

[Ci15] H. Y. Kim, “A New Public-Key Authentication Watermarking for Binary Document Images Resistant to Parity Attacks,” in *Proc. IEEE Int. Conf. on Image Processing*, vol. 2, pp. 1074-1077, 2005.

[Ci14] H. Y. Kim and R. L. de Queiroz, “Alteration-Locating Authentication Watermarking for Binary Images,” *Int. Workshop on Digital Watermarking 2004,* (Seoul), Lecture Notes in Computer Science 3304, pp. 125-136, 2004.

[Ci13] H. Y. Kim and R. L. Queiroz, “A Public-Key Authentication Watermarking for Binary Images,” in *Proc. IEEE Int. Conf. on Image Processing*, (Singapore), pp. 3459-3462, 2004.

Artigos sobre watermarking reversível para imagens binárias:

[Cn21] S. V. D. Pamboukian and H. Y. Kim, “Reversible Data Hiding and Reversible Authentication Watermarking for Binary Images,” in *Proc. Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg)*, 2006.

Artigo sobre imagem que aparece sobrepondo duas imagens. Também como determinar a máscara de difusão de erro usada:

[Pei2003] 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.

Artigo sobre watermarking de imagens binárias alterando um único pixel

[Chang2005] Chin-Chen Chang, Chun-Sen Tseng and Chia-Chen Lin, "Hiding Data in Binary Images," Lecture Notes in Computer Science, Volume 3439/2005, Pages 338-349, 2005.