Selective encryption of uncompressed and compressed images

M. Van Droogenbroeck

November 6, 2002

Outline

- Motivation
- Discussion on steganography and hiding capacity
- Discussion on compression and encryption
- 2 techniques of selective encryption
- Multiple selective encryption

Motivation

Several tools for several functionalities:

- Steganography $\Longrightarrow$ Information hiding

??? Data hiding

- Watermarking $\Longrightarrow$ Management of “Digital Rights”

- Encryption $\Longrightarrow$ Confidentiality, integrity, signature, ...

Motivation

Several tools for several functionalities:

- Steganography $\Longrightarrow$ Information hiding

??? Data hiding

- Watermarking $\Longrightarrow$ Management of “Digital Rights”

- Encryption $\Longrightarrow$ Confidentiality, integrity, signature, ...

Example of a real application:

- How to securely access an image database?

Steganography, data hiding

Definition [Data hiding in images] Data hiding techniques are processes that embed data, such as copyright information, with a minimum amount of degradation to the host signal, called cover.

Amongst others,

- MASRY et al. (2001) proposed a technique based on a wavelet decomposition.
- HONSINGER et al. (1999) and FRIDRICH et al. (2002) have proposed lossless data embedding techniques that allow the recovery of the original image.

Data-hiding capacity?

- M. BARNI, F. BARTOLINI et al. (1999). Capacity of the Watermark-Channel: How Many Bits Can Be Hidden Within a Digital Image?

- P. MOULIN and M. MIHÇAK (2002). A Framework for Evaluating the Data-Hiding Capacity of Image Sources. Trans. on Image Processing, September 2002.

Data-hiding capacity?

- M. BARNI, F. BARTOLINI et al. (1999). Capacity of the Watermark-Channel: How Many Bits Can Be Hidden Within a Digital Image?

- P. MOULIN and M. MIHÇAK (2002). A Framework for Evaluating the Data-Hiding Capacity of Image Sources. Trans. on Image Processing, September 2002.

Conclusions

- Data-hiding capacity is content dependent

- Data-hiding capacity $<$ 0.5 bit/pixel

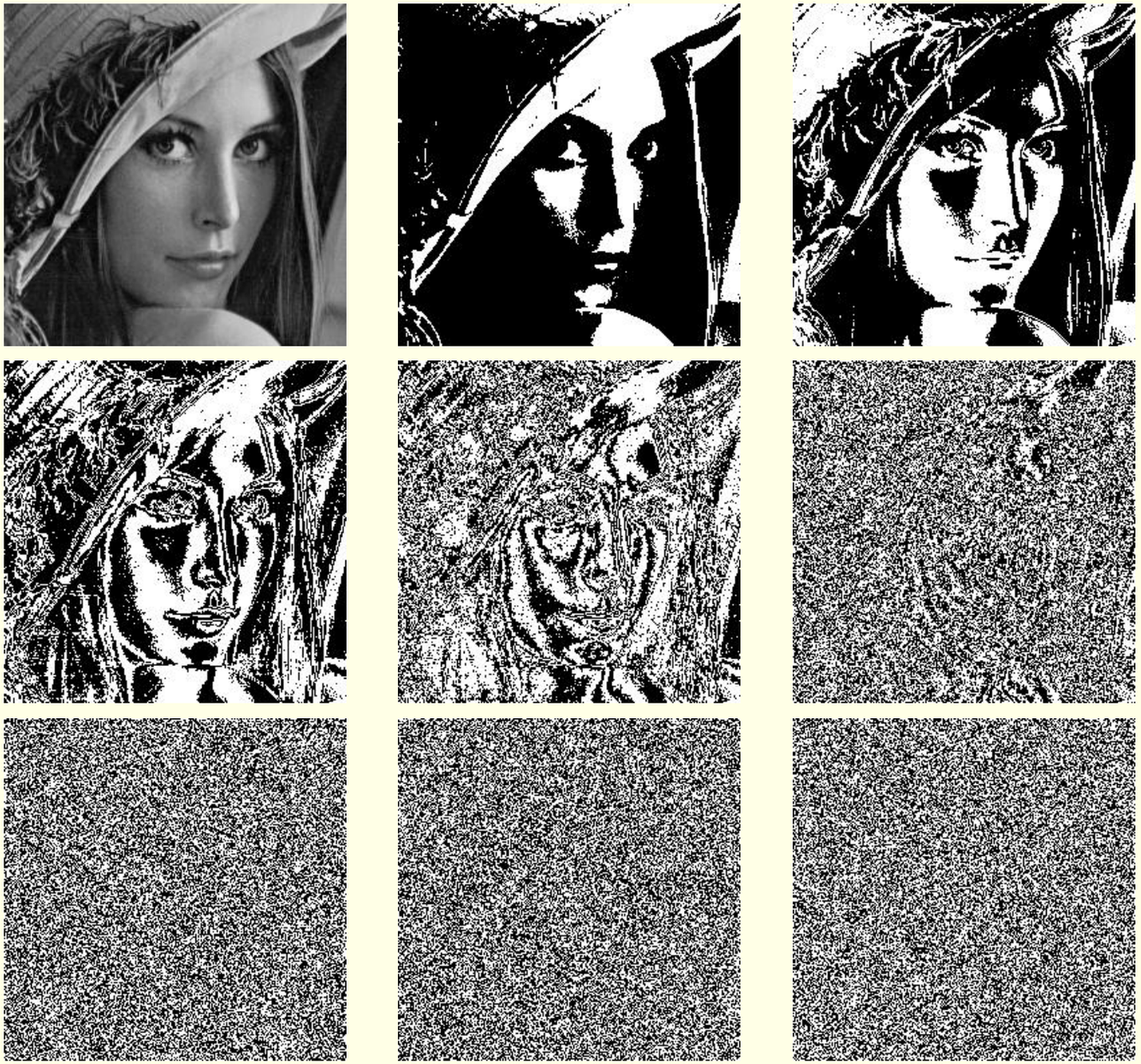
Remark that they consider the hidden bits as the important information. But is this true?

Properties of a data hiding method?

- The modified image is subjectively identical to the original image.
- The data is directly embedded in the image, rather than into a header.
- The embedded data is self-detectable. This means that there is no need to put any specific information in the header or elsewhere.
- The embedding technique allows the inclusion of a maximum amount of information with respect to the image content.
- The embedded information offers a service to the user. Therefore there is no reason to remove or alter the embedded information.

These properties are similar to the properties of feature location techniques as defined by BENDER et al. (1996).

LENA and her bitplanes starting from the most significant bit



Naive method: embedding information in the Least Significant Bits (LSBs)

Technique: embed the information in the least significant bits of each pixel value by substitution.

- Estimation of the error $\Longrightarrow$ leading to some threshold, depending on the amount of information to be embedded.

Calculation of the error

Notations:

- $o(x, y)$, $c(x, y)$ denotes the original image and the modified image.
- An error function is defined as $e(x, y) = o(x, y) - c(x, y)$
  - Decomposition in bitplanes:

$$e(x,y) = o(x,y) - c(x,y) = (o_0 - c_0) + 2(o_1 - c_1) + 4(o_2 - c_2) + \ldots = e_0 + 2e_1 + 4e_2 + \ldots$$

4 scenarios:

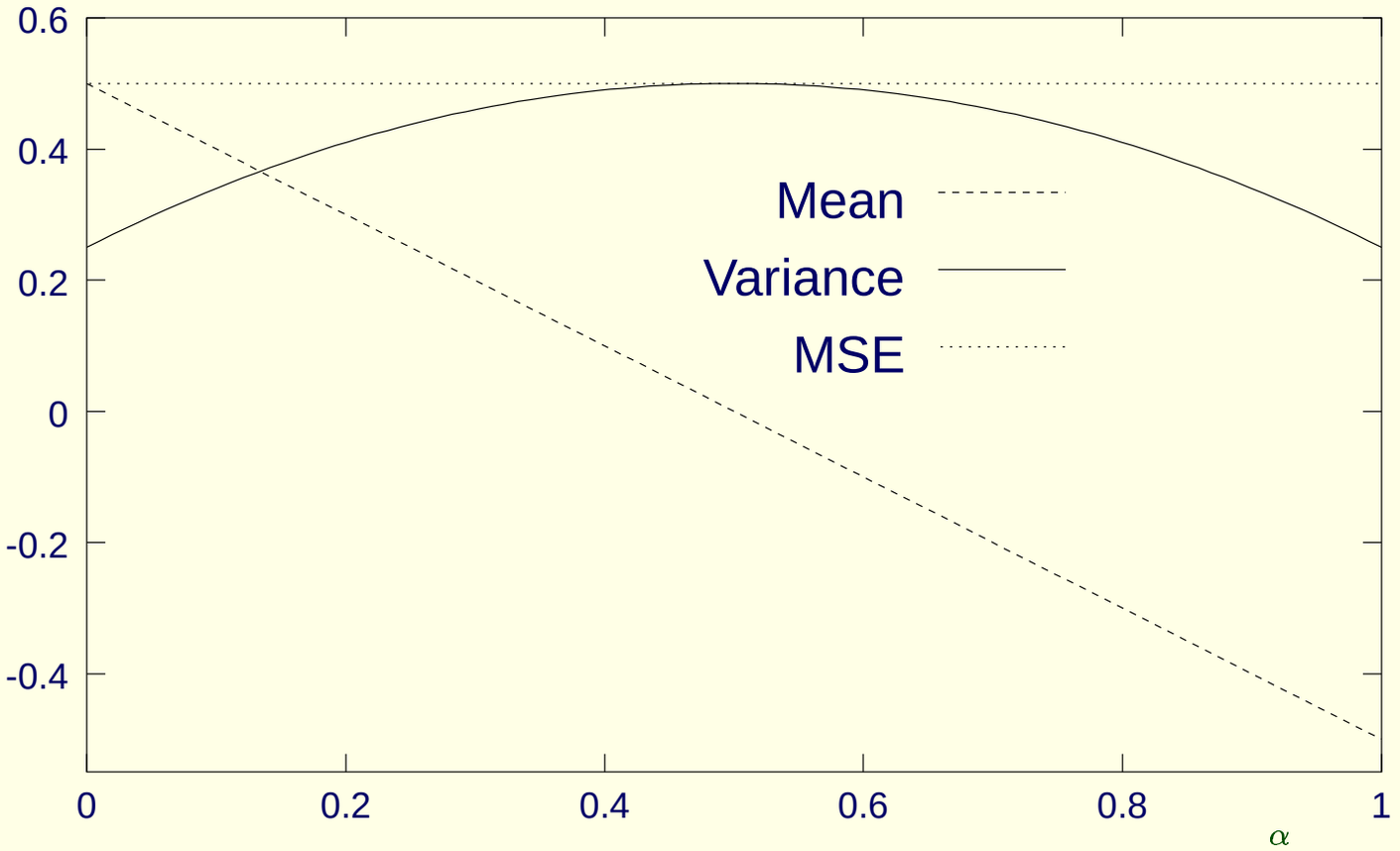
	$c_k = 0$	$c_k = 1$
$o_k = 0$	$e_k = 0$	$e_k = -1$
$o_k = 1$	$e_k = +1$	$e_k = 0$

Bit probabilities?

- In a bitplane of the original image, $p_o(0) = \alpha$ and $p_o(1) = 1 - \alpha$.
- For the modified data, $p_e(0)$ = $p_e(1)$ = $\frac{1}{2}$
  - With the assumptions of stationarity, the mean and variance of e_k (the error in bitplane k) are equal to:

$$\mu_{e_k} = \frac{1}{2} - \alpha, \qquad \sigma^2_{e_k} = -\alpha^2 + \alpha + \frac{1}{4} \tag{1}$$

μ_{e_k}, $\sigma^2_{e_k}$ and MSE_k as functions of α



Derivation of σ_e^2 from the $\sigma_{e_k}^2$

$$e(x,y) = o(x,y) - c(x,y) = (o_0 - c_0) + 2(o_1 - c_1) + 4(o_2 - c_2) + \ldots = e_0 + 2e_1 + 4e_2 + \ldots$$

If we assume the independence between the bitplane values,

$$\sigma_e^2 = \sigma_{e_0}^2 + 4\sigma_{e_1}^2 + 16\sigma_{e_2}^2 + \ldots = \sum_{k=0}^{n-1} 2^{2k} \left(-\alpha_k^2 + \alpha_k + \frac{1}{4} \right)$$

where n is the number of least significant bits that have been substituted.

Derivation of σ_e^2 from the $\sigma_{e_k}^2$

$$e(x,y) = o(x,y) - c(x,y) = (o_0 - c_0) + 2(o_1 - c_1) + 4(o_2 - c_2) + \ldots = e_0 + 2e_1 + 4e_2 + \ldots$$

If we assume the independence between the bitplane values,

$$\sigma_e^2 = \sigma_{e_0}^2 + 4\sigma_{e_1}^2 + 16\sigma_{e_2}^2 + \ldots = \sum_{k=0}^{n-1} 2^{2k} \left(-\alpha_k^2 + \alpha_k + \frac{1}{4} \right)$$

where n is the number of least significant bits that have been substituted.

When $p_o(0) = p_o(1) = \frac{1}{2}$, which is a valid assumption on the least significant bits of a natural image,

$$\mu_e = 0$$

$$\sigma_e^2 = \frac{4^n - 1}{3} \times \frac{1}{2} = \frac{4^n - 1}{6} = \mathrm{MSE}$$

Description of an entropy based embedding technique

Our embedding technique is based on the modification of the least significant bits. The algorithm adapts the number of embedded bits to the image content according to the following steps:

1. the image is divided in 8×8 pixels wide blocks.

2. for each 8×8 block, denoted B, compute the entropy $H(B)$ on the 4 most significant bits. If the entropy is larger than 2, then embed information in the 4 least significant bits of B, else go to step 3.

3. for each block compute the entropy $H(B)$ on the 5 most significant bits. If the entropy is larger than 2 then embed information in the 3 least significant bits of B, else embed information in the 2 least significant bits of B.

Properties:

- Because the entropy computed during step 2 is based on the 4 most significant bits, there are only 16 possible values and therefore the entropy is contained in the $[0, 4]$ interval.

- The entropy computed during step 3 is different from that of step 2 as the computation is based on 5 bits; accordingly the entropy is contained in $[0, 5]$.

- In all cases, the algorithm provides a minimum of 2 embedded bits per pixel. But in the best case, the number may raise to 4 bits.
  Compare this to the data-hiding capacity of 0.5 bit/pixel !

At the reception side?

A similar algorithm is used to extract the embedded message:

1. the image is divided in 8×8 pixels wide blocks.

2. for each 8×8 block, compute the entropy $H(B)$ on the 4 most significant bits. If the entropy is larger than 2, then retrieve the 4 least significant bits of B, else go to step 3.

3. for each block compute the entropy $H(B)$ on the 5 most significant bits. If the entropy is larger than 2 then retrieve the 3 least significant bits of B, else retrieve the 2 least significant bits of B.

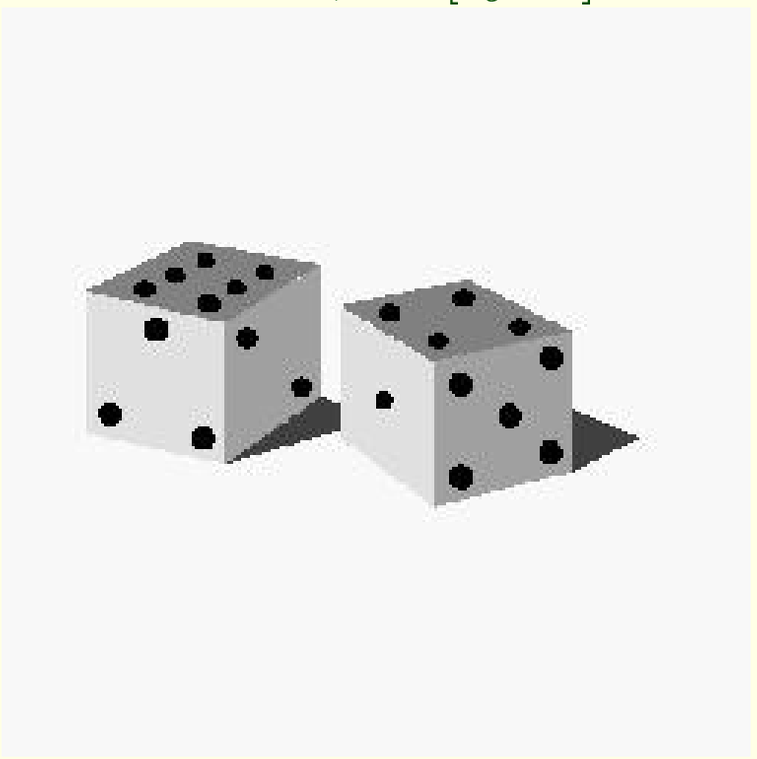
Lena $65,536\ [bytes]$

$H(C) = 7.52$

Message = $25,048\ [bytes]$

Message = $3.06\ [bit/pixel]$

Dice $65,536\ [bytes]$



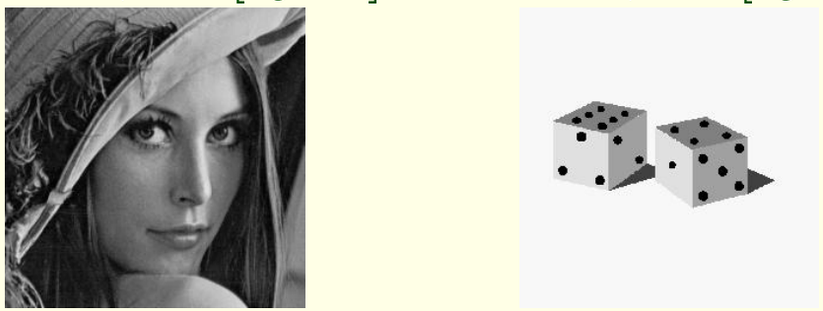
$H(C) = 0.96$

Message = $16,384\ [bytes]$

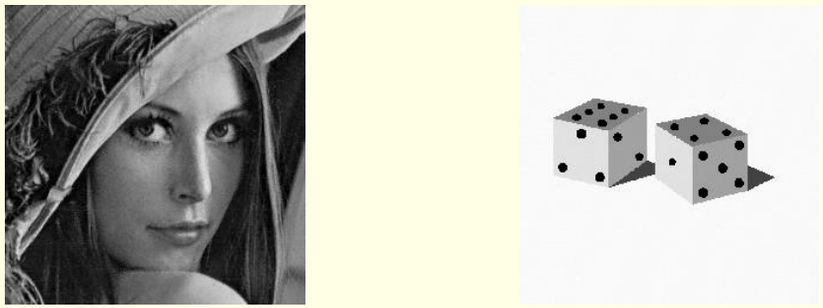
Message = $2\ [bit/pixel]$

Figure 1: Images and sizes of the message that can be embedded.

Lena $65,536\ [bytes]$ Dice $65,536\ [bytes]$



Lena + message Dice + message



Error function Error function

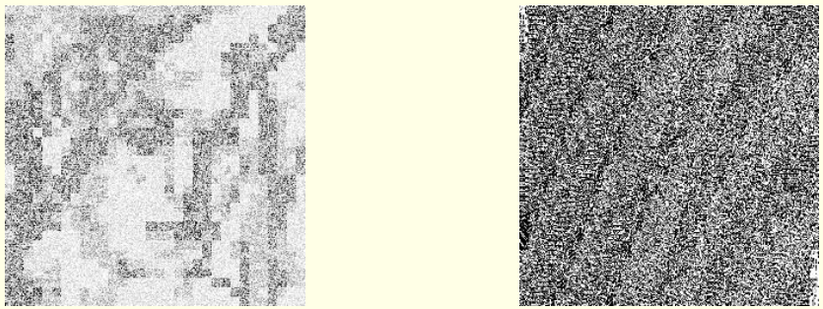


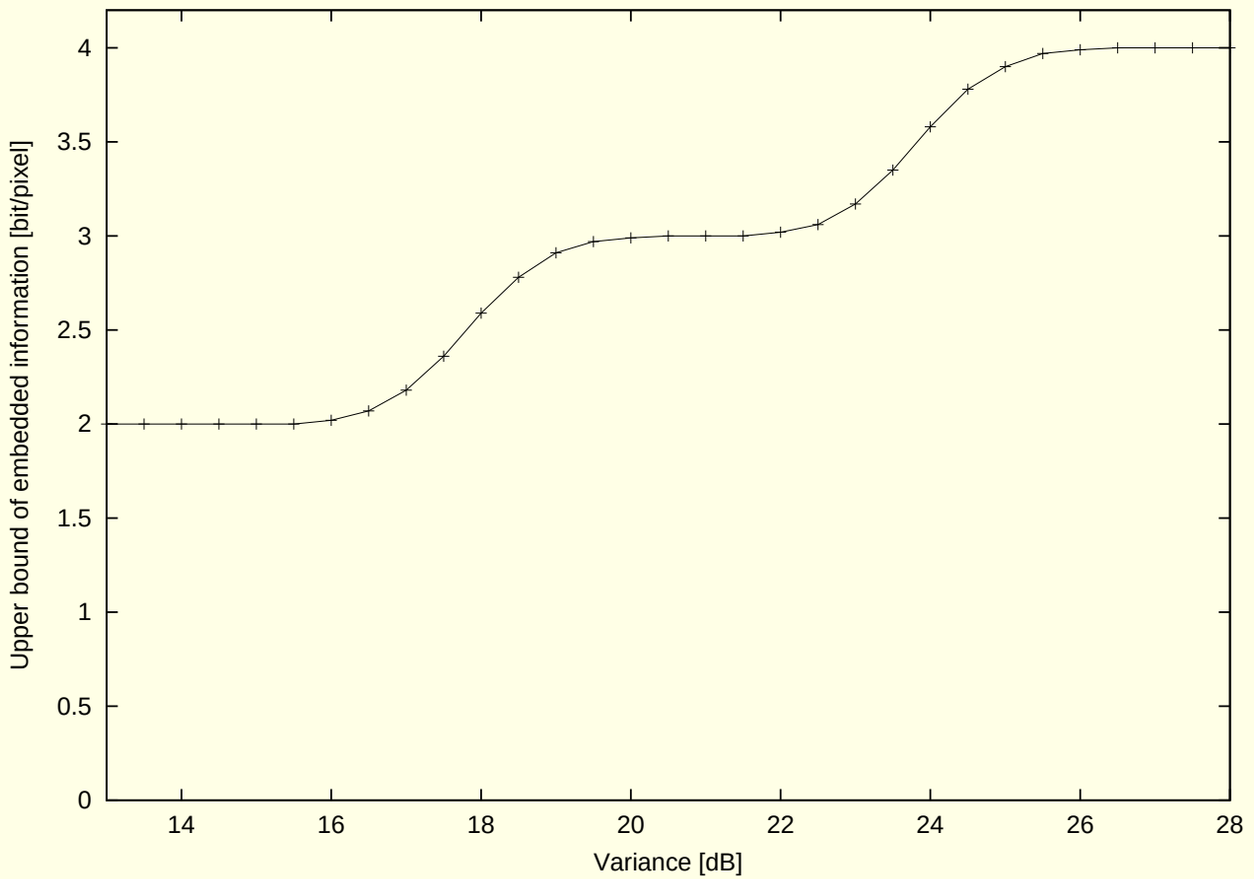
Figure 2: Illustrations of our embedding technique.

Determination of the upper bound of information that can be embedded

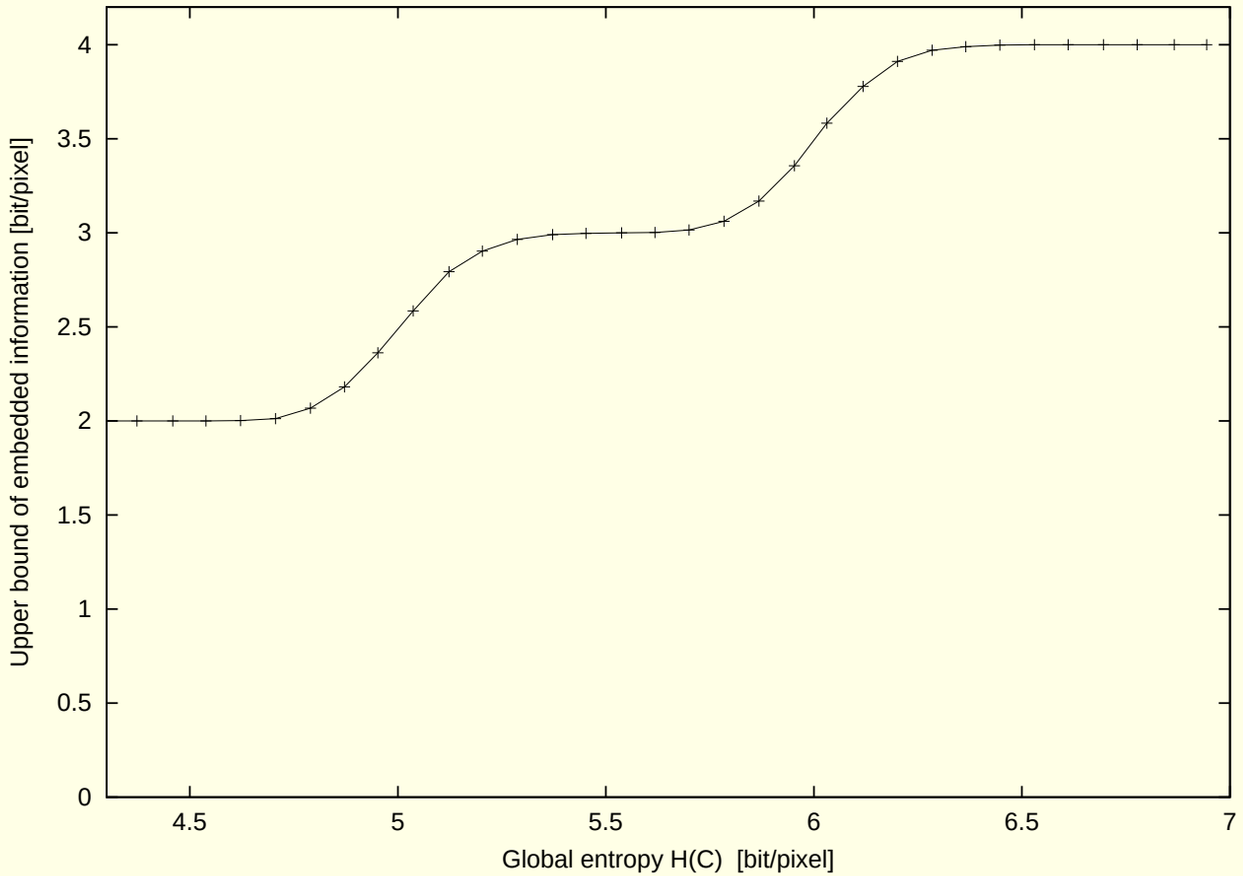
Real applications require to know in advance the number of bits that can be embedded in an image.

To provide an upper bound, we used an image in which values were generated by a gaussian process $N(127.5, \sigma_N^2)$ centered on 127.5. All values were rounded to the closest integer in $[0, 255]$.

Link between σ_N^2 and the average upper bound of embedded bits per pixel



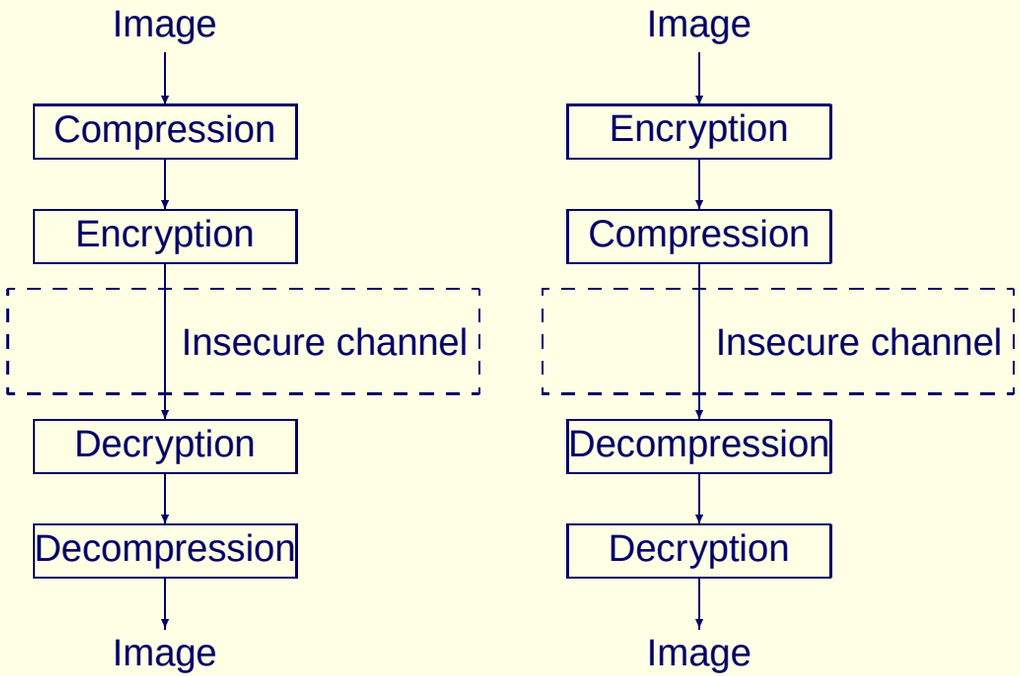
Link between the entropy of the cover $H(C)$ and the average upper bound of embedded bits per pixel



Outline

- Motivation
- Discussion on steganography and hiding capacity
- ▷ Discussion on compression and encryption
- 2 techniques of selective encryption
- Multiple selective encryption

Compression or encryption first?



Combining compression and encryption

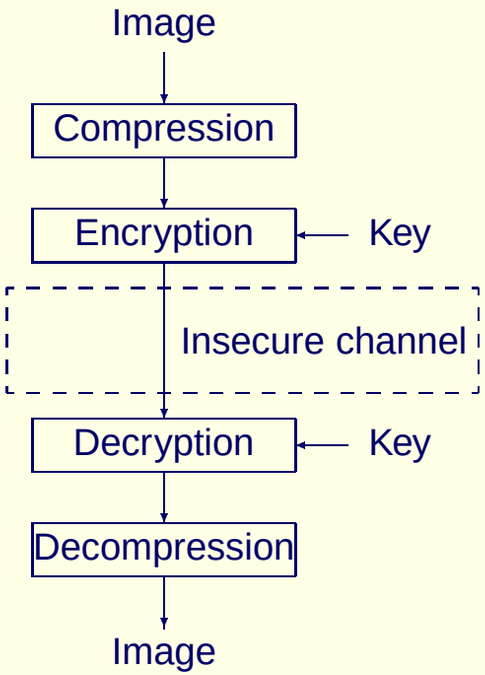
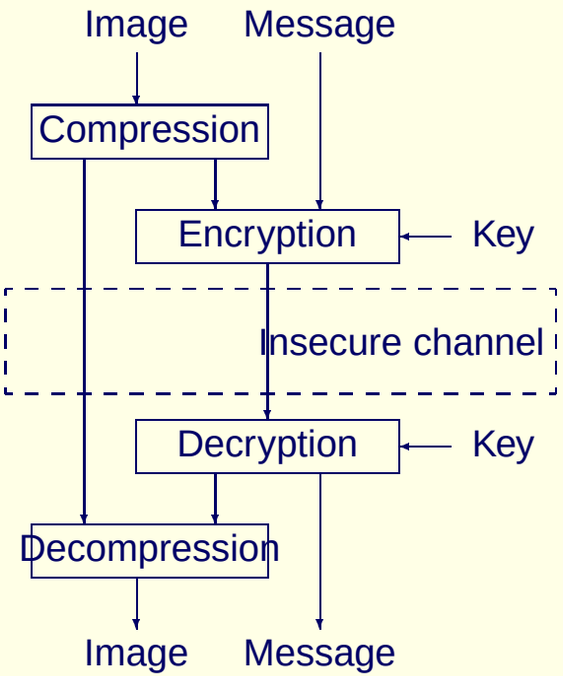


Figure 3: Encryption of an image.

- The encryption process requires an encryption algorithm and a key
- The encryption algorithm should be published (formalized in KIRKHOFF's law)

Selective encryption mechanism



When the decryption key is unknown

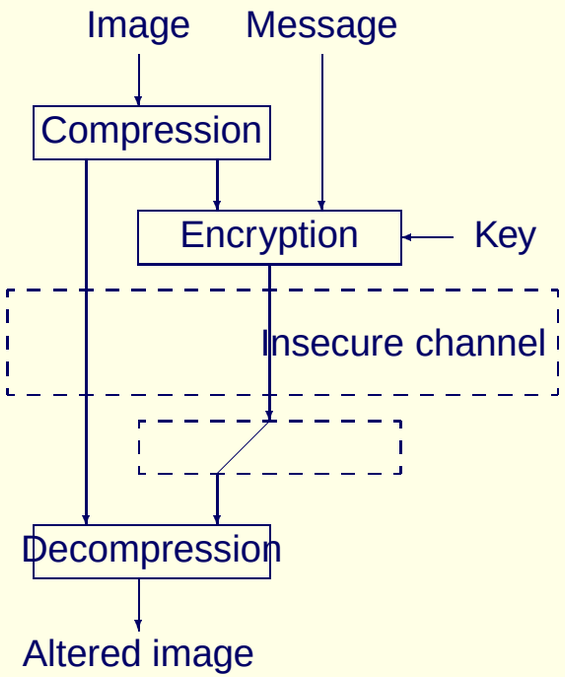


Figure 4: When the decryption key is unknown to the receiver.

Selective encryption of uncompressed images

Description of a “naive” method:

- a XOR function is sufficient when the message is only used once
- encrypt each bitplane separately and reconstruct a gray level image

LENA and her bitplanes starting from the most significant bit

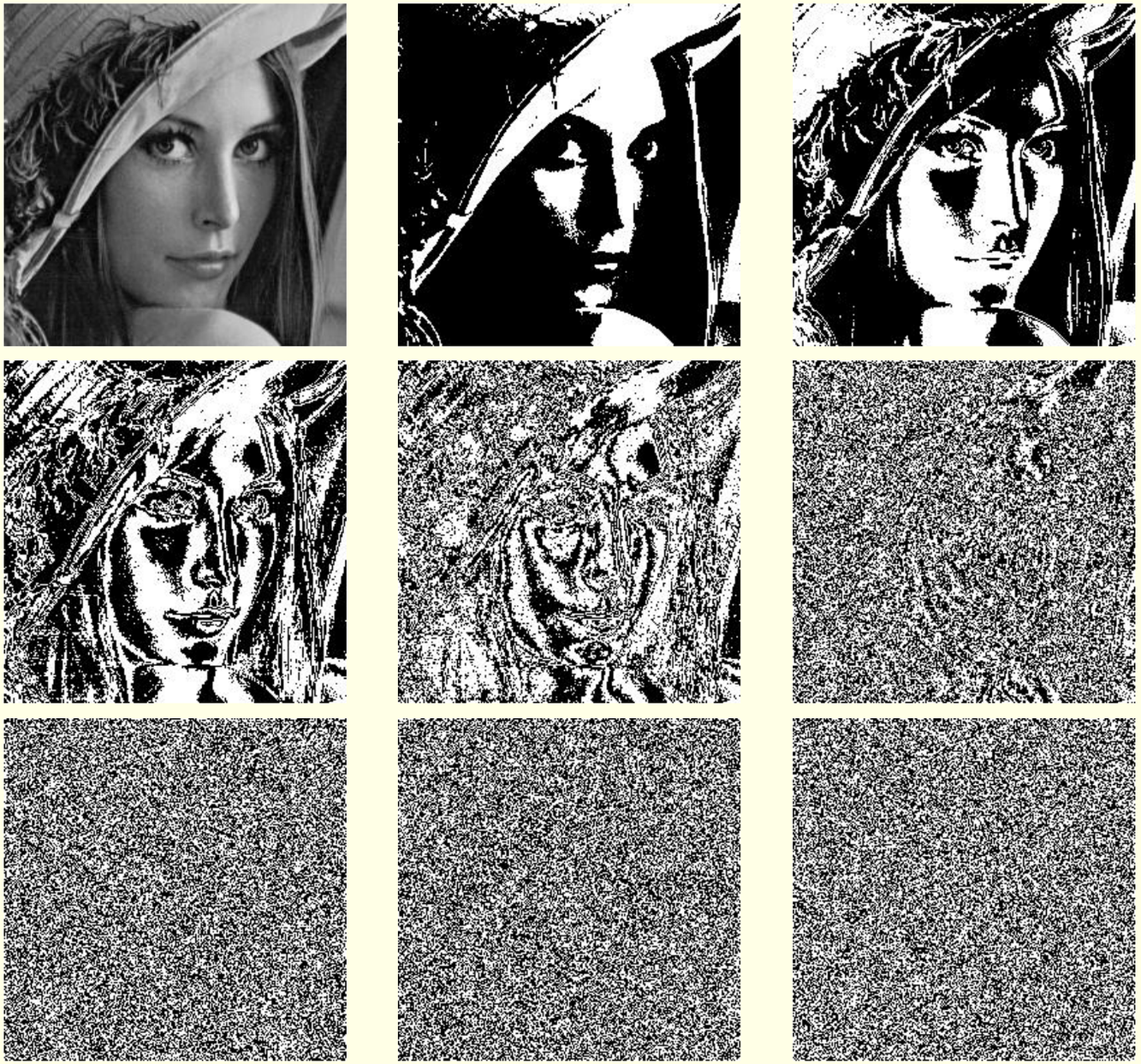


Illustration of a naive selective encryption method

(a) Original image

(b) 3 bits encrypted

$MSE = 10.6$

$PSNR = 37.9\,[dB]$

(c) 5 bits encrypted

$MSE = 171$

$PSNR = 25.8\,[dB]$

(d) 7 bits encrypted

$MSE = 2704$

$PSNR = 13.8\,[dB]$

Derivation of σ_e^2 from the $\sigma_{e_k}^2$

Reminder:

When $p_o(0) = p_o(1) = \frac{1}{2}$, which is a valid assumption on the least significant bits of a natural image,

$$\mu_e = 0$$

$$\sigma_e^2 = \frac{4^n - 1}{3} \times \frac{1}{2} = \frac{4^n - 1}{6} = \mathrm{MSE}$$

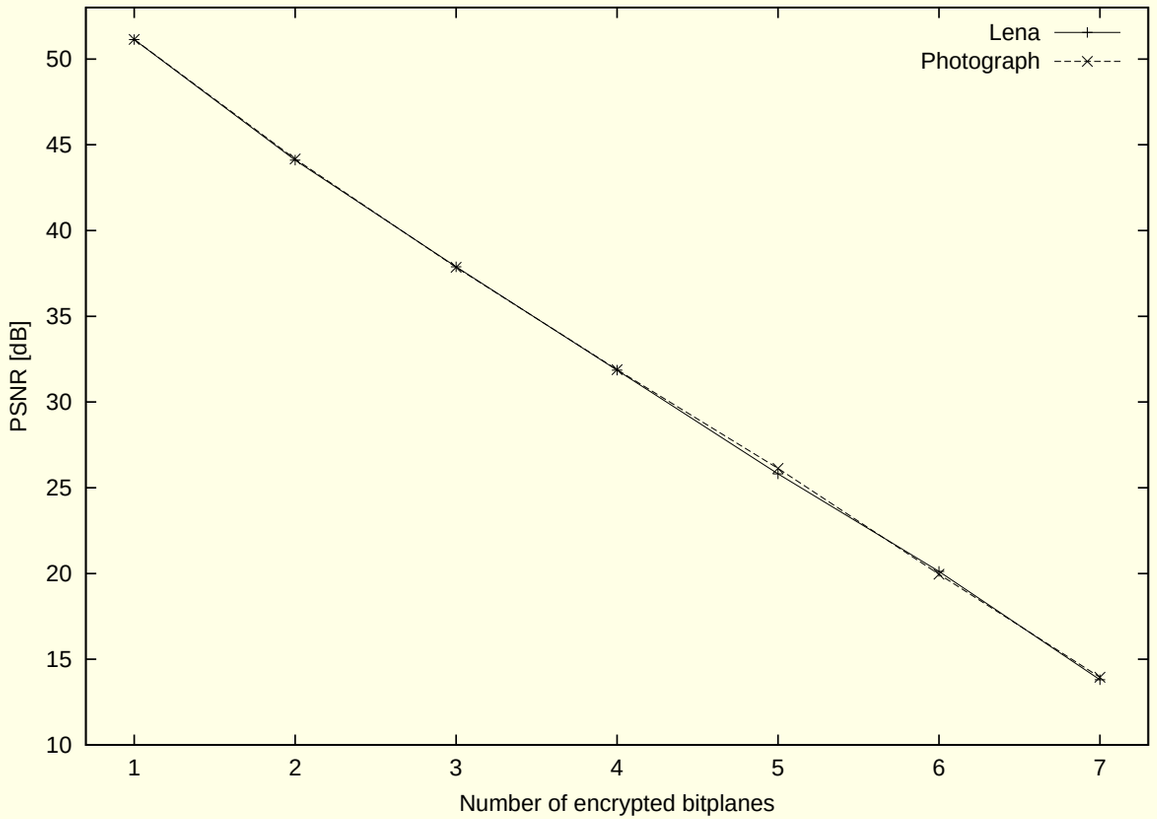
Comparison of theoretical MSE values with computed values on LENA and PHOTOGRAPH

$\sharp$	Theoretical MSE	MSE_{Lena}	$MSE_{Photograph}$
1	0.5	0.5	0.5
2	2.5	2.5	2.5
3	10.5	10.6	10.6
4	42.5	42.6	42.5
5	170.5	171	158.4
6	682.5	636.7	654.3
7	2730.5	2714.8	2584.7

Table 1: Comparison of MSEs for a number $\sharp$ of encrypted bitplanes.

PSNR values versus the number of encrypted bitplanes

$$10 \log \sigma_e^2 \approx 10n \log 4 - 10 \log 6 \tag{2}$$



Demonstrations

- 1 to 7 bits encrypted video stream
- Encrypted motion blocks

Selective encryption of compressed images

Requirements:

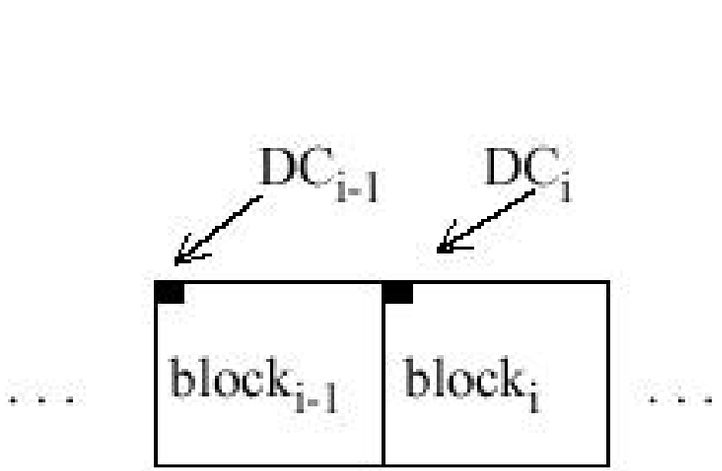
[Visual acceptance] part of information may be visible but the encrypted image should look noisy,

[Selective encryption] encryption occurs after compression and leaves parts of the bitstream unencrypted,

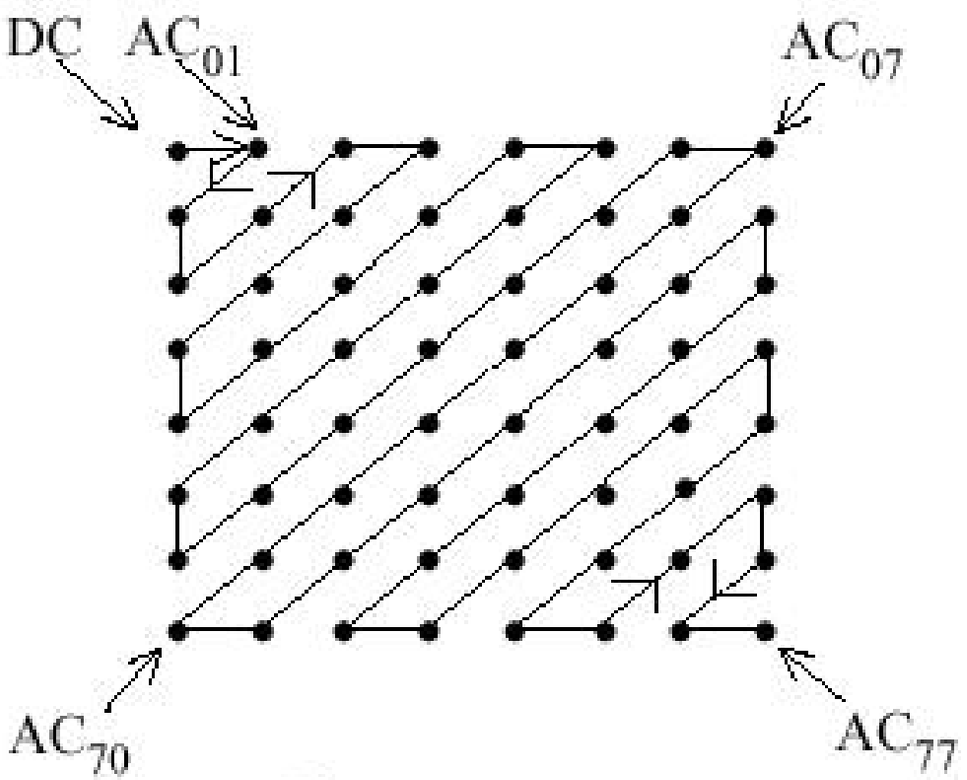
[Constant bit rate] encryption should preserve the size of the bitstream, and

[Bitstream compliance] the encryption step should produce a compliant bitstream according to the chosen format definition.

A method for the selective encryption of JPEG images



Differential DC encoding



Zigzag sequence

Coding of AC coefficients

symbol-1 (RUNLENGTH, SIZE)	symbol-2 (AMPLITUDE)

SIZE	AMPLITUDE
1	-1,1
2	-3,-2,2,3
3	-7..-4,4..7
4	-15..-8,8..15
5	-31..-16,16..31
6	-63..-32,32..63
7	-127..-64,64..127
8	-255..-128,128..255
9	-511..-256,256..511
10	-1023..-512,512..1023

Illustration

Two cases: all coefficients are encrypted except (a) the DC coefficient or (b) the DC, AC_0, $\ldots$, AC_4 coefficients.

(a) DC
$MSE = 1096$
$PSNR = 17.7\,[dB]$

(b) DC, AC_0, $\ldots$, AC_4
$MSE = 348$
$PSNR = 22.7\,[dB]$

Figure 5: JPEG encrypted images.

Image quality

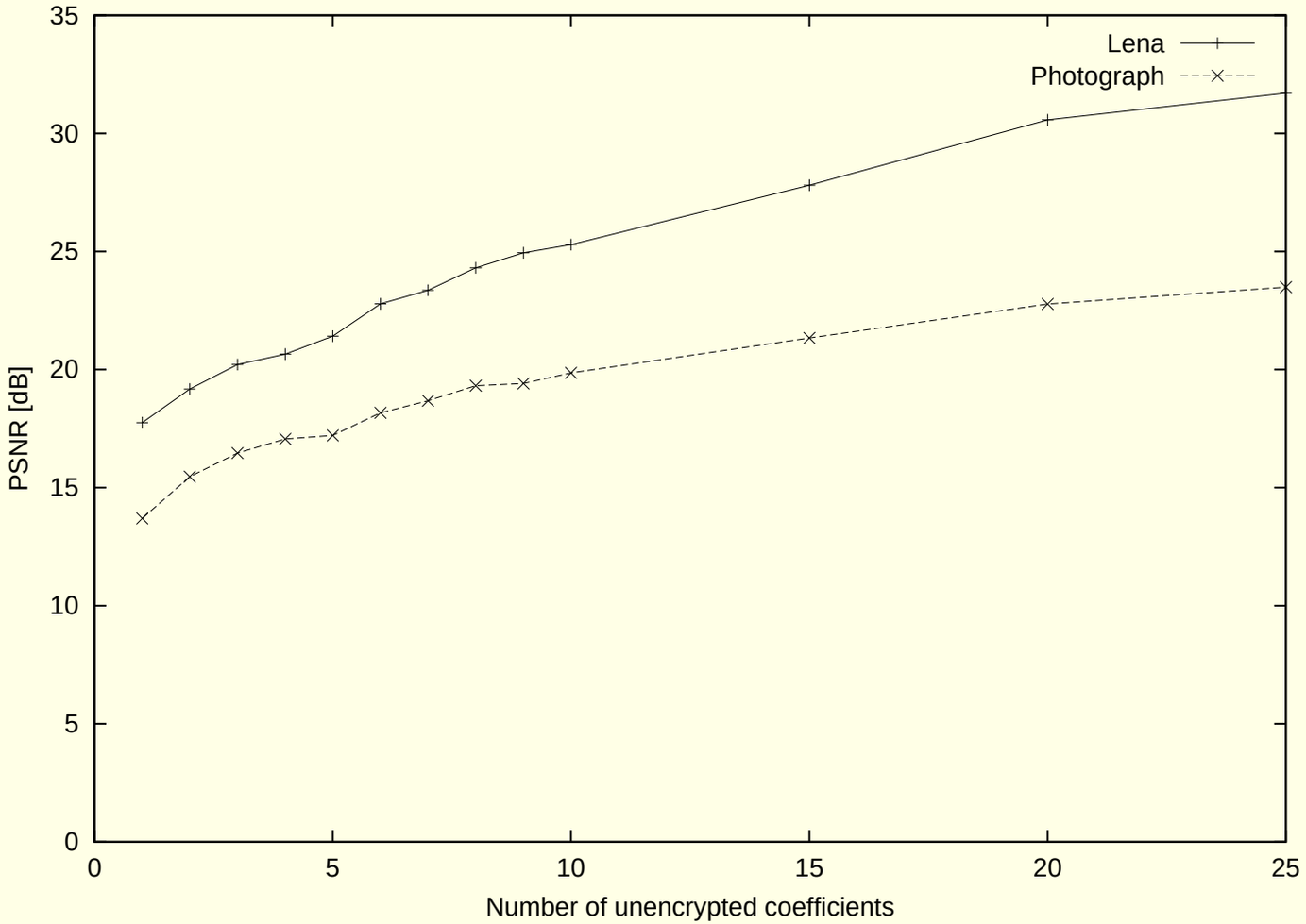


Figure 6: PSNR values versus the number of coefficients left unencrypted (including the DC coefficient).

Performances

When the encryption does not occur inside the coder, the encryption steps are:

- read the JPEG bitstream,
- build the Huffman tables as specified in the image,
- extract DCT coefficients,
- proceed to encryption, and
- replace the bits in the bitstream.

Computation speed

	Encrypted bits		Execution speed [ms]		
bpp	total	per bloc	DES	3-DES	IDEA
3.62	275904	22.45	77	82	75
2.14	135744	11.05	45	48	44
1.01	52864	8.6	19	20	19
0.49	13760	2.24	8	9	8

Table 2: Execution times in [ms] for LENA (512×512, 24-bit color image) on a 1.33 GHz Pentium

Multiple selective encryption

A first user applies the selective DCT encryption algorithm on the JPEG image f with a key $k1$:

$$g = E_{k1}(f)$$

The decryption algorithm D:

$$f = D_{k1}(E_{k1}(f))$$

Multiple selective encryption

A first user applies the selective DCT encryption algorithm on the JPEG image f with a key $k1$:

$$g = E_{k1}(f)$$

The decryption algorithm D:

$$f = D_{k1}(E_{k1}(f))$$

If there is a second user with his own key $k2$:

$$h = E_{k2}(E_{k1}(f))$$

We introduce the term of "multiple selective encryption".

Multiple selective encryption

A first user applies the selective DCT encryption algorithm on the JPEG image f with a key $k1$:

$$g = E_{k1}(f)$$

The decryption algorithm D:

$$f = D_{k1}(E_{k1}(f))$$

If there is a second user with his own key $k2$:

$$h = E_{k2}(E_{k1}(f))$$

We introduce the term of "multiple selective encryption".

A better method is called "selective over-encryption" :

$$E_{k1}(D_{k2}(E_{k1}(f)))$$

Illustration of techniques called multiple selective encryption (b) and selective over-encryption (c-d)

(a) Original JPEG image

(b) $E_{k2}(E_{k1}(f))$

(c) $E_{k1}(D_{k2-3}(E_{k1}(f)))$

(d) $E_{k1}(D_{k2-2}(E_{k1}(f)))$