

Digital Steganography: *Hiding Data within Data*

Hide and Seek: An Introduction to Steganography

Donovan Artz

Niels Provos & Peter Honeyman



Meet at the front of the trap

random capitalosis is a rare disease often contracted by careless internet users. this sad illness causes the affected person to randomly capitalize letters in a body of text. please do not confuse this disease with a blatant attempt at steganography.

Figure 1. Difficulty hiding data. This text, encoded as 8-bit ASCII, is 254 bytes long. The hidden message, at 23 bytes, is about ten times smaller than the carrier data but remains conspicuous.



An encoding example

Television static
An essay of pictures

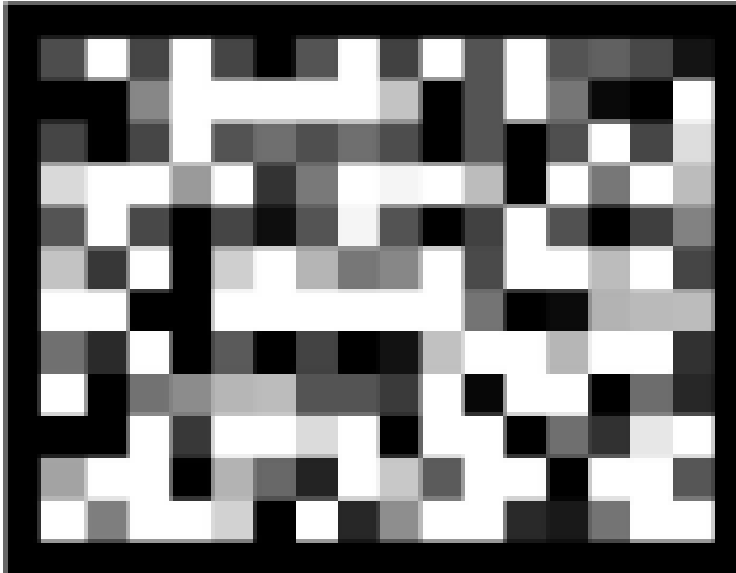


Figure 2. A medium conducive to success. The message from Figure 1 can be hidden using a picture of television static. The static in the black box can be encoded in 192 bytes, which is only eight times larger than the hidden message.

Excluding the black border, start at the pixel in the upper left corner of the image.

Set the color value of the current pixel to the ASCII value of the corresponding character in the message you want to hide.

Move two pixels to the right. If you are at the edge of the image, wrap around and skip a line.

Repeat the previous two steps until the entire message is coded.



Methods for Digital Steganography

- Images as Carriers
- Audio File Carriers
- Data Ordering

HTML

```
<IMG SRC="il.jpg" ALT="image 1" NAME="i1">  
<IMG SRC="il.jpg" NAME="i1" ALT="image 1">
```

Perl

```
$d = $a + $b + $c;  
$d = $a + $c + $b;  
$d = $b + $a + $c;  
. . .
```

E-mail

```
to: dono@drerel.edu,  
dono@lanl.gov  
to: dono@lanl.gov,  
dono@drerel.edu
```

Figure 4. Examples of steganography using permutations. HTML and PERL offer steganographic potential, but steganography using data ordering can be limited by the number of permutations possible.



```
C:\WINDOWS\System32\cmd.exe

Z:\Development\MP3Stego>encode -E hidden_text.txt -P pass svega.wav svega_stego.mp3
MP3StegoEncoder 1.1.15
See README file for copyright info
Microsoft RIFF, WAUVE audio, PCM, mono 44100Hz 16bit, Length: 0: 0:20
MPEG-I layer III, mono Psychoacoustic Model: AT&T
Bitrate=128 kbps De-emphasis: none CRC: off
Encoding "svega.wav" to "svega_stego.mp3"
Hiding "hidden_text.txt"
[Frame 791 of 791] (100.00%) Finished in 0: 0: 6

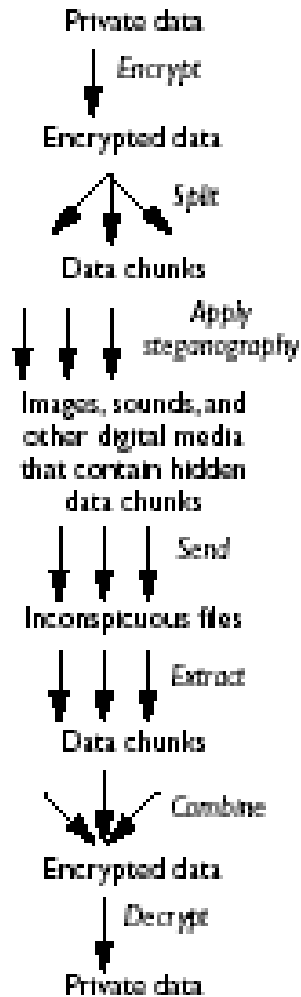
Z:\Development\MP3Stego>decode -X -P pass svega_stego.mp3
MP3StegoEncoder 1.1.15
See README file for copyright info
Input file = 'svega_stego.mp3' output file = 'svega_stego.mp3.pcm'
Will attempt to extract hidden information. Output: svega_stego.mp3.txt
the bit stream file svega_stego.mp3 is a BINARY file
HDR: s=FFF, id=1, l=3, ep=off, br=9, sf=0, pd=1, pr=0, m=3, js=0, c=0, o=0, e=0
alg.=MPEG-1, layer=III, tot bitrate=128, sfrq=44.1
mode=single-ch, sblim=32, jsbd=32, ch=1
[Frame 791] Avg slots/frame = 417.434; b/smp = 2.90; br = 127.839 kbps
Decoding of "svega_stego.mp3" is finished
The decoded PCM output file name is "svega_stego.mp3.pcm"

Z:\Development\MP3Stego>_
```

- compresses svega.wav (mono, 44.1 kHz, 16bit encoded) and hides hidden_text.txt. The hidden text is encrypted using pass as a password. This produces the output called [svega_stego.mp3](#). If no information was hidden, you would obtain [this](#).
- uncompresses svega_stego.mp3 into svega_stego.mp3.pcm and attempts to extract hidden information. The hidden message is decrypted, uncompressed and saved into svega_stego.mp3.txt



Example Process



- **Steganography and encryption**
 - separate goals
- **Limitations**
 - shared secrets

Figure 3. Process flow of secretly transmitting data. Using steganography, private data remains undetectable until it reaches its intended audience.



Steganography Sites

Steganography Sites

Data Hiding Homepage

<http://nif.www.media.mit.edu/DataHiding/>

Information Hiding homepage

<http://www.cl.cam.ac.uk/~fapp2/steganography/>

Steganalysis

<http://www.jjtc.com/Steganalysis/>

Steganography and Digital Watermarking

<http://www.jjtc.com/Steganography/>

StegoArchive.com

<http://steganographytripod.com/stego.html>

Watermarking Mailing List

<http://www.watermarkingworld.org/ml.html>



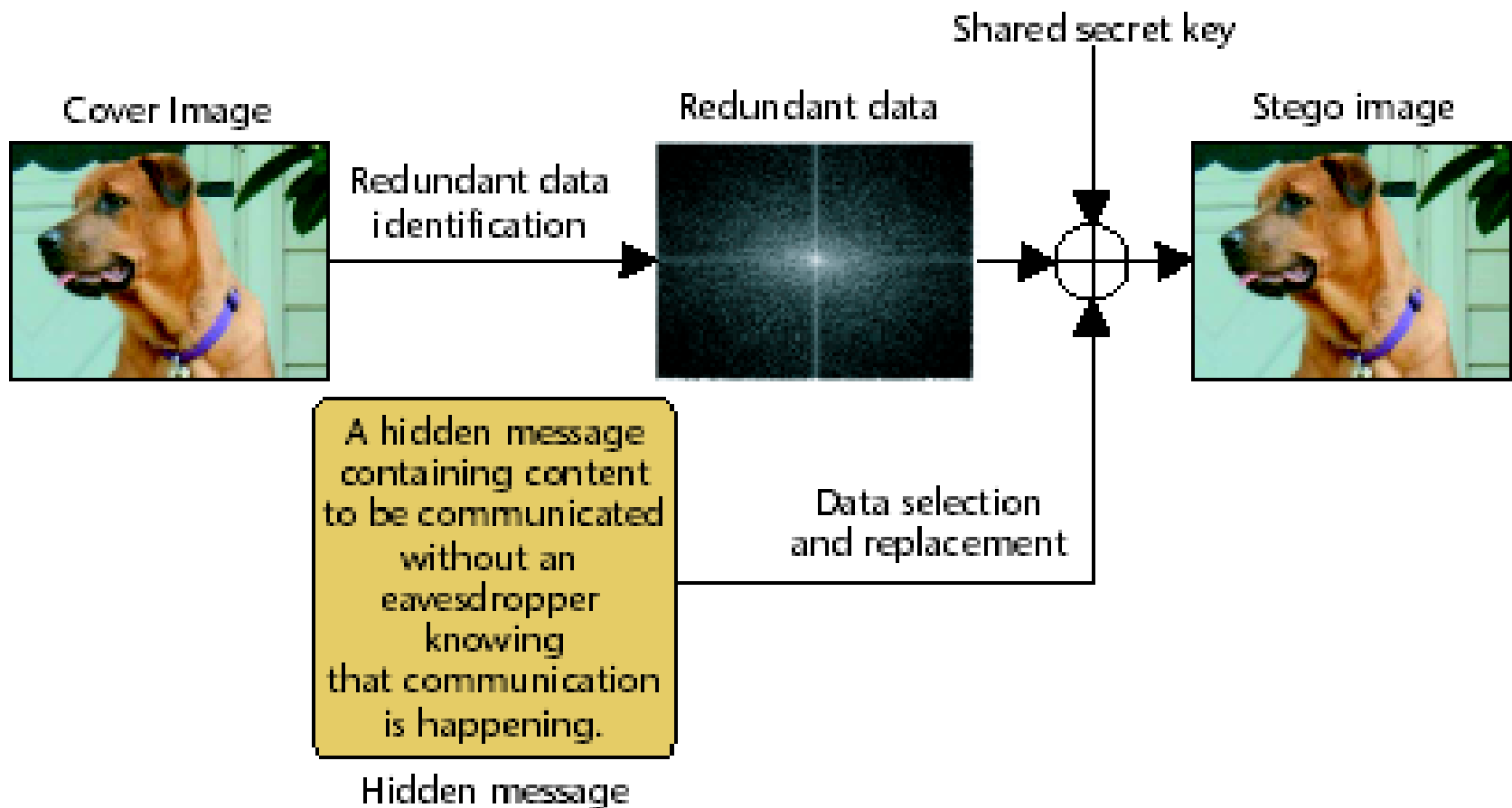


Figure 1. Modern steganographic communication. The encoding step of a steganographic system identifies redundant bits and then replaces a subset of them with data from a secret message.

Discrete Cosine Transform

For each color component, the JPEG image format uses a *discrete cosine transform* (DCT) to transform successive 8×8 pixel blocks of the image into 64 DCT coefficients each. The DCT coefficients $F(u, v)$ of an 8×8 block of image pixels $f(x, y)$ are given by

$$F(u, v) = \frac{1}{4} C(u) C(v) \left[\sum_{x=0}^7 \sum_{y=0}^7 f(x, y) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16} \right],$$
$$F^Q(u, v) = \left\lfloor \frac{F(u, v)}{Q(u, v)} \right\rfloor,$$

where $Q(u, v)$ is a 64-element quantization table.

where $C(x) = 1/\sqrt{2}$ when x equal 0 and $C(x) = 1$ otherwise. Afterwards, the following operation quantizes the coefficients:



Sequential

In the simple case, the embedding step changes the least-significant bit of colors in an image. The colors are addressed by their indices i in the color table; we refer to their respective frequencies before and after embedding as n_i and n_i^* . Given uniformly distributed message bits, if $n_{2i} > n_{2i+1}$, then pixels with color $2i$ are changed more frequently to color $2i + 1$ than pixels with color $2i + 1$ are changed to color $2i$. As a result, the following relation is likely to hold:

$$|n_{2i} - n_{2i+1}| \geq |n_{2i}^* - n_{2i+1}^*|.$$

In other words, embedding uniformly distributed message bits reduces the frequency difference between adjacent colors.



Frequency Histograms

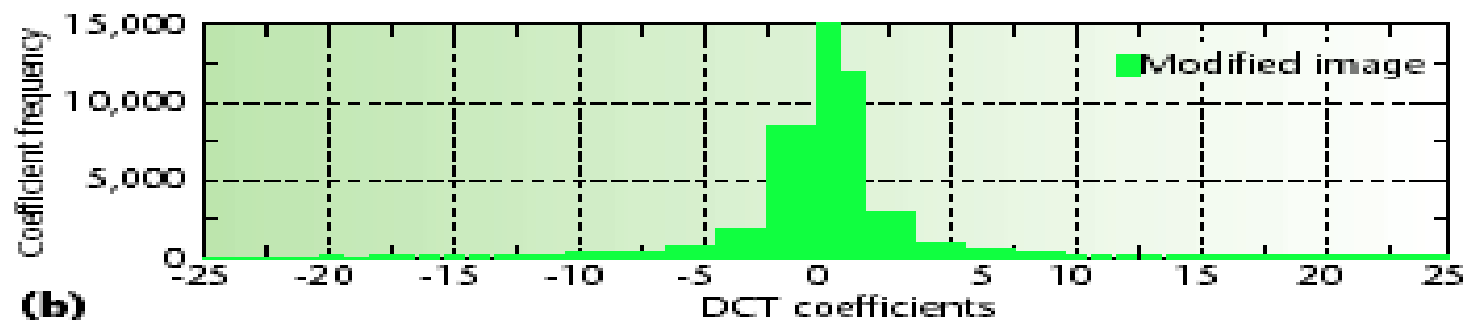
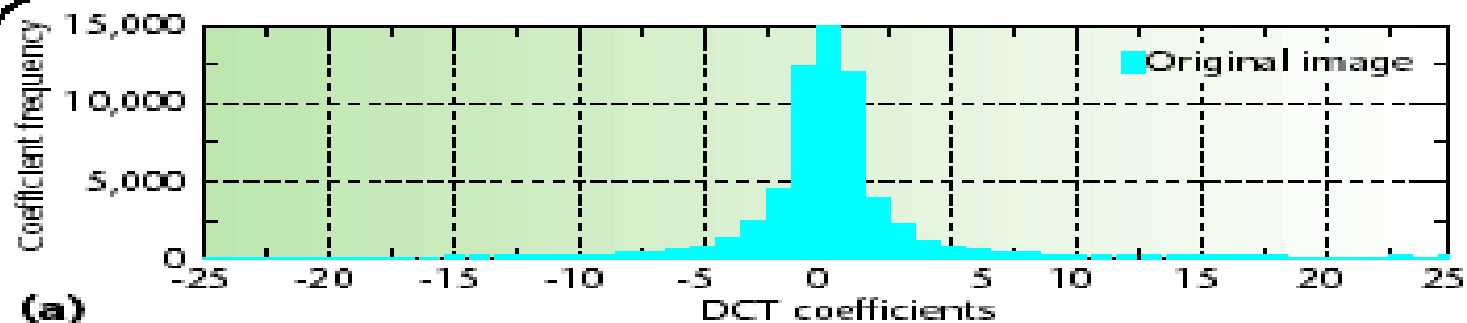


Figure 4. Frequency histograms. Sequential changes to the (a) original and (b) modified image's least-sequential bit of discrete cosine transform coefficients tend to equalize the frequency of adjacent DCT coefficients in the histograms.

Probability of Embedding

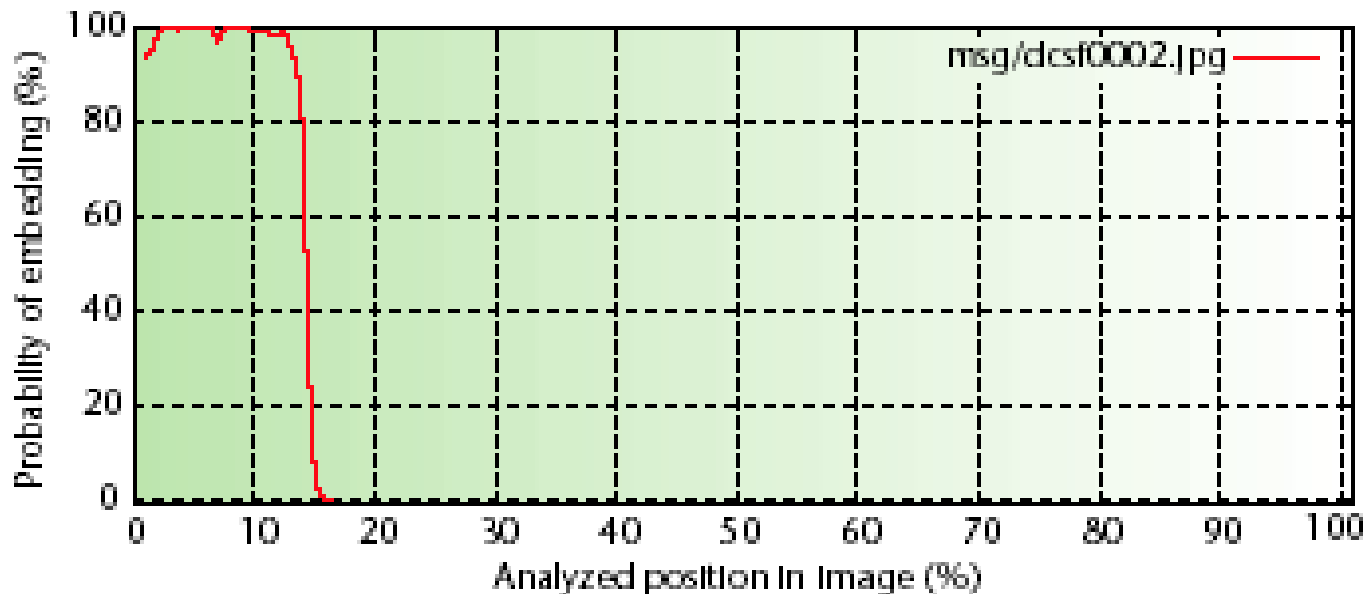


Figure 5. A high probability of embedding indicates that the image contains steganographic content. With JSteg, it is also possible to determine the hidden message's length.



Pseudo-Random (Outguess)

```
Input: message, shared secret, cover image
Output: stego image
initialize PRNG with shared secret
while data left to embed do
    get pseudo-random DCT coefficient from cover image
    if DCT  $\neq$  0 and DCT  $\neq$  1 then
        get next LSB from message
        replace DCT LSB with message LSB
    end if
    insert DCT into stego image
end while
```

Figure 6. The OutGuess 0.1 algorithm. As it runs, the algorithm replaces the least-significant bit of pseudo-randomly selected discrete cosine transform (DCT) coefficients with message data.



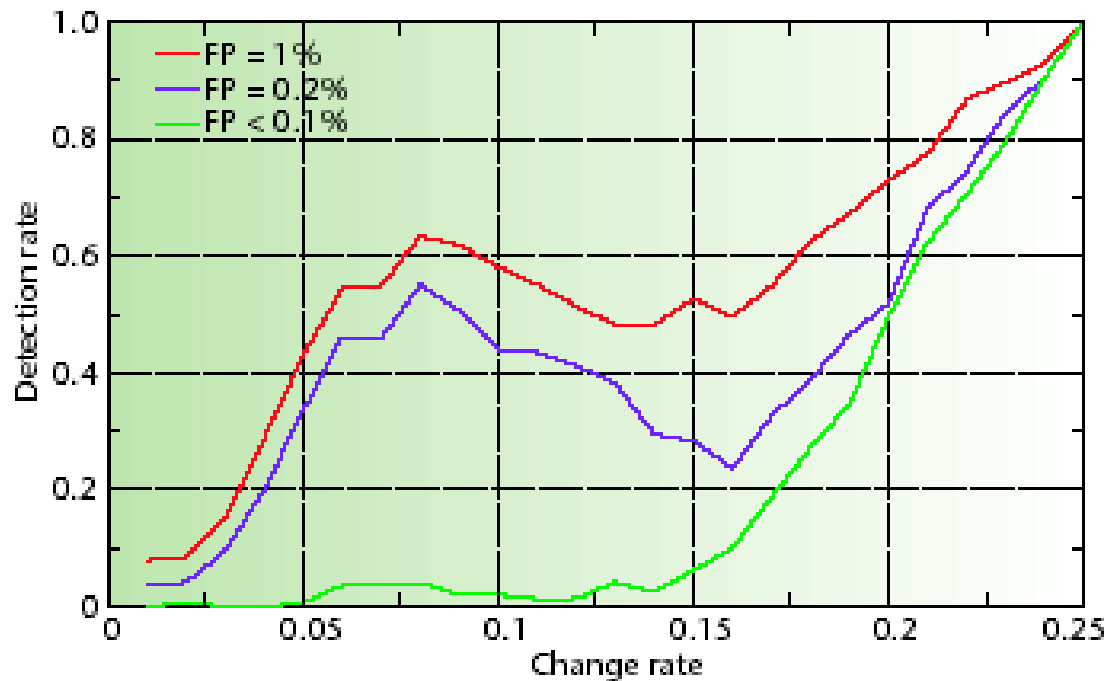


Figure 7. The extended χ^2 -test detects pseudo-randomly embedded messages in JPEG images. The detection rate depends on the hidden message's size and can be improved by applying a heuristic that eliminates coefficients likely to lead to false negatives. The graph shows the detection rates for three different false-positive rates. The change rate refers to the fraction of discrete cosine transform (DCT) coefficients available for embedding a hidden message that have been modified.

Detection Rate

Table 1. Detection rate P_D for a nonlinear support vector machine.¹¹

SYSTEM	MESSAGE IMAGE SIZE	P_D IN PERCENT	
		(P_F 1.0)	(P_F 0.0)
JSteg	256 × 256	99.0	98.5
JSteg	128 × 128	99.3	99.0
JSteg	64 × 64	99.1	98.7
JSteg	32 × 32	86.0	74.5
OutGuess	256 × 256	95.6	89.5
OutGuess	128 × 128	82.2	63.7
OutGuess	64 × 64	54.7	32.1
OutGuess	32 × 32	21.4	7.2

1



OutGuess and F5 Detection

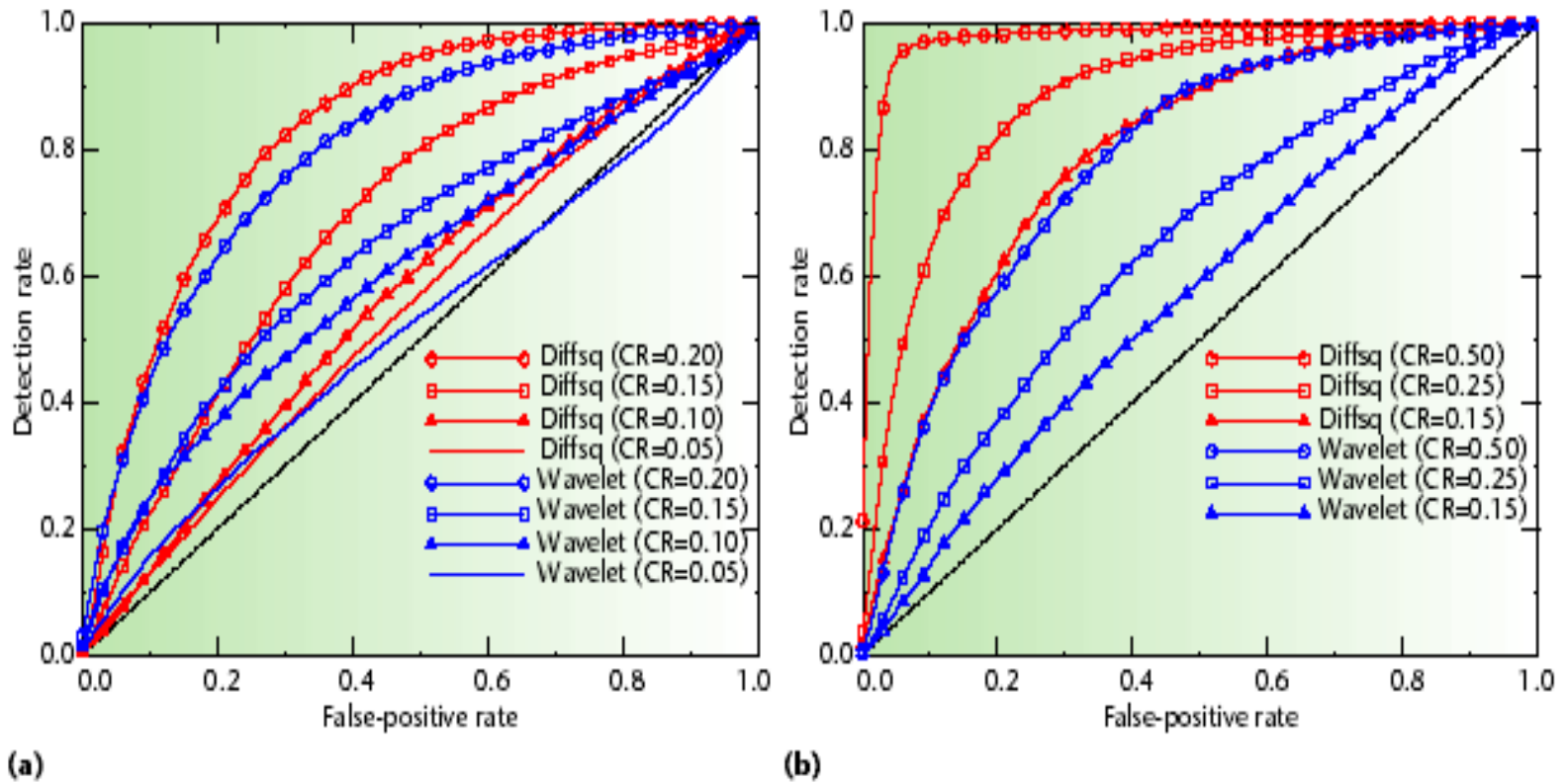


Figure 8. Different feature vectors based on wavelet-like decomposition and on squared differences. (a) The receiver operating characteristic (ROC) for OutGuess detection and (b) the ROC for F5 detection.

F5 Algorithm

```
Input: message, shared secret, cover image
Output: stego image
initialize PRNG with shared secret
permute DCT coefficients with PRNG
determine  $k$  from image capacity
calculate code word length  $n \leftarrow 2^k - 1$ 
while data left to embed do
    get next  $k$ -bit message block
    repeat
         $G \leftarrow \{n \text{ non-zero AC coefficients}\}$ 
         $s \leftarrow k\text{-bit hash } f \text{ of LSB in } G$ 
         $s \leftarrow s \oplus k\text{-bit message block}$ 
        if  $s \neq 0$  then
            decrement absolute value of DCT coefficient  $G_s$ 
            insert  $G_s$  into stego image
        end if
    until  $s = 0$  or  $G_s \neq 0$ 
    insert DCT coefficients from  $G$  into stego image
end while
```

Figure 9. The F5 algorithm. F5 uses subtraction and matrix encoding to embed data into the discrete cosine transform (DCT) coefficients.



Receiver-operating characteristics (F5)

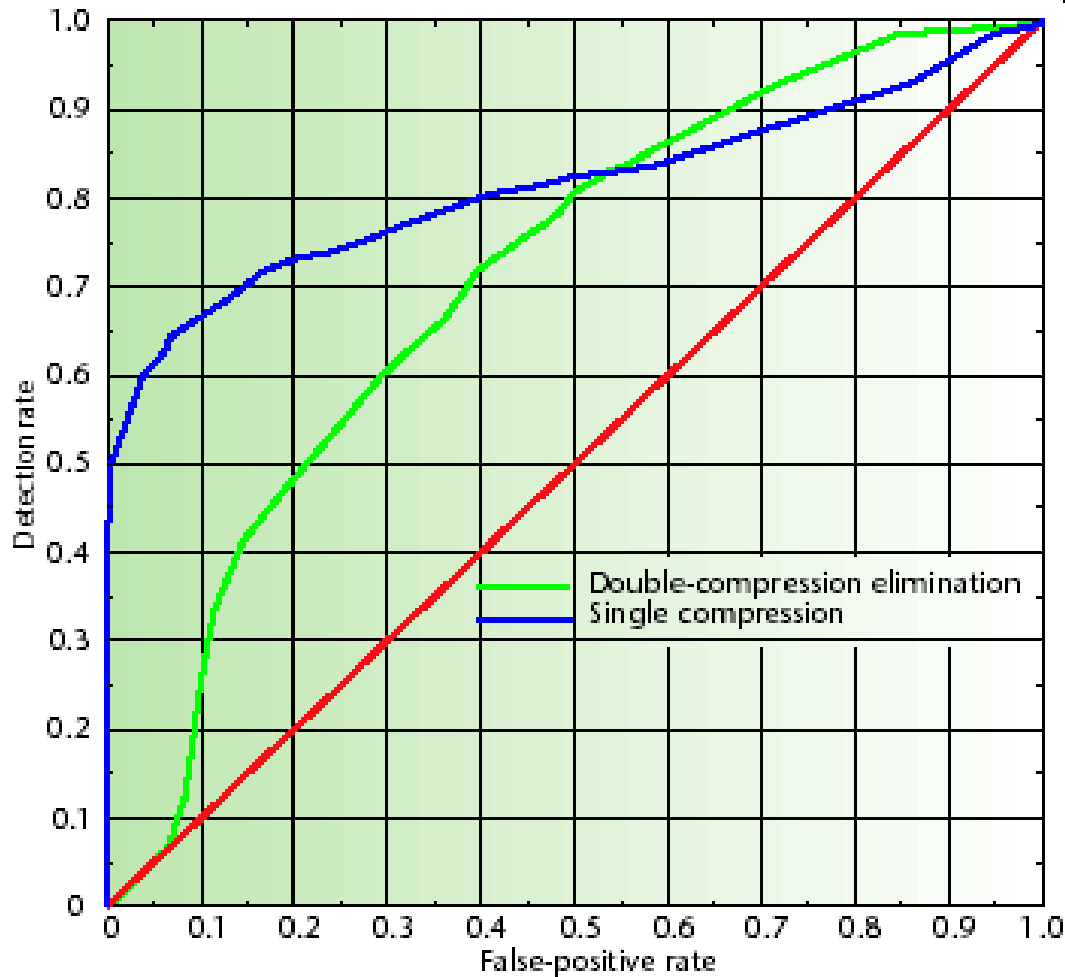
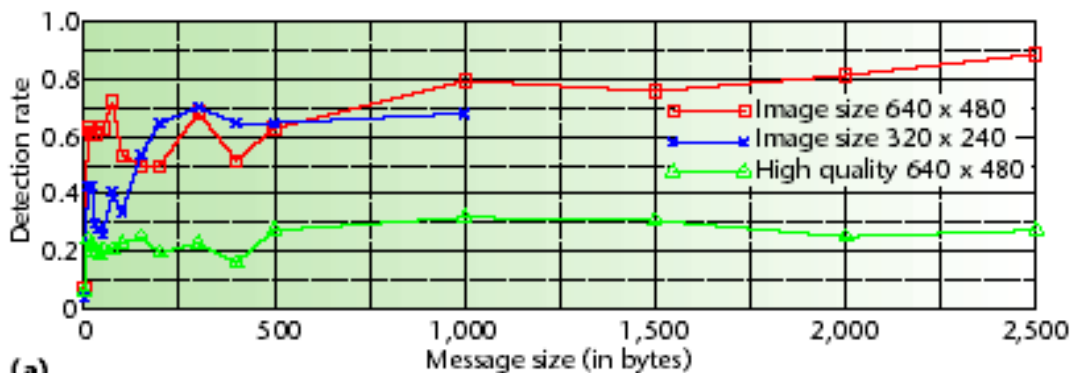


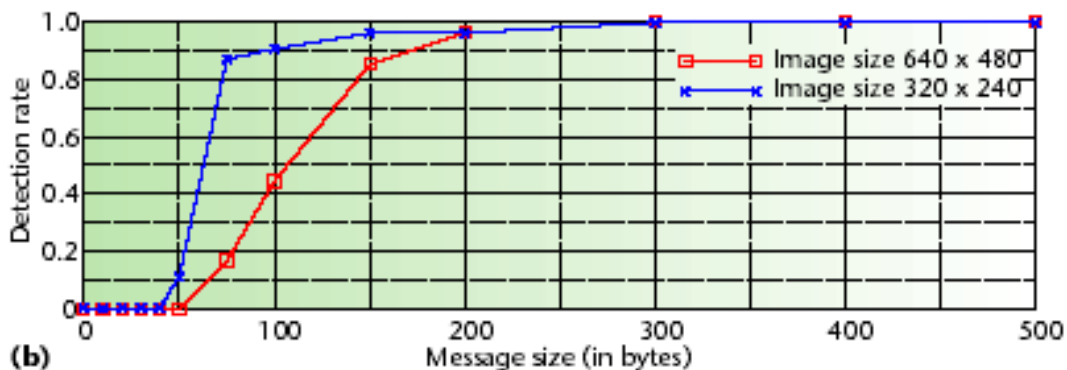
Figure 10. Receiver-operating characteristics (ROCs) of the F5 detection algorithm. The detection rate is analyzed when using double compression elimination and against single compressed images.



Stegdetect



(a)



(b)

Figure 11. Using Stegdetect over the Internet. (a) JPHide and (b) JSteg produce different detection results for different test images and message sizes.



False Positives

Table 2. Percentages of (false) positives for analyzed images.

TEST	EBAY	USENET
JSteg	0.003	0.007
JPHide	1	2.1
OutGuess	0.1	0.14



StegBreak Performance

Table 3. Stegbreak performance on a 1,200-MHz Pentium III.

SYSTEM	ONE IMAGE (WORDS/SECOND)	FIFTY IMAGES (WORDS/SECOND)
JPHide	4,500	8,700
OutGuess 0.13b	18,000	34,000
JSteg	36,000	47,000

