Hide and Seek: An Introduction to Steganography

Although people have hidden secrets in plain sight—now called steganography—throughout the ages, the recent growth in computational power and technology has propelled it to the forefront of today's security techniques.

Niels Provos and Peter Honeyman
University of Michigan

Steganography is the art and science of hiding communication; a steganographic system thus embeds hidden content in unremarkable cover media so as not to arouse an eavesdropper's suspicion. In the past, people used hidden tattoos or invisible ink to convey steganographic content. Today, computer and network technologies provide easy-to-use communication channels for steganography.

Essentially, the information-hiding process in a steganographic system starts by identifying a cover medium's redundant bits (those that can be modified without destroying that medium's integrity).[1] The embedding process creates a *stego medium* by replacing these redundant bits with data from the hidden message.

Modern steganography's goal is to keep its mere presence undetectable, but steganographic systems—because of their invasive nature—leave behind detectable traces in the cover medium. Even if secret content is not revealed, the existence of it is: modifying the cover medium changes its statistical properties, so eavesdroppers can detect the distortions in the resulting stego medium's statistical properties. The process of finding these distortions is called *statistical steganalysis*.

This article discusses existing steganographic systems and presents recent research in detecting them via statistical steganalysis. Other surveys focus on the general usage of information hiding and watermarking or else provide an overview of detection algorithms.[2,3] Here, we present recent research and discuss the practical application of detection algorithms and the mechanisms for getting around them.

The basics of embedding

Three different aspects in information-hiding systems contend with each other: capacity, security, and robustness.[4] Capacity refers to the amount of information that can be hidden in the cover medium, security to an eavesdropper's inability to detect hidden information, and robustness to the amount of modification the stego medium can withstand before an adversary can destroy hidden information.

Information hiding generally relates to both watermarking and steganography. A watermarking system's primary goal is to achieve a high level of robustness—that is, it should be impossible to remove a watermark without degrading the data object's quality. Steganography, on the other hand, strives for high security and capacity, which often entails that the hidden information is fragile. Even trivial modifications to the stego medium can destroy it.

A classical steganographic system's security relies on the encoding system's secrecy. An example of this type of system is a Roman general who shaved a slave's head and tattooed a message on it. After the hair grew back, the slave was sent to deliver the now-hidden message.[5] Although such a system might work for a time, once it is known, it is simple enough to shave the heads of all the people passing by to check for hidden messages—ultimately, such a steganographic system fails.

Modern steganography attempts to be detectable only if secret information is known—namely, a secret

key.[2] This is similar to Kerckhoffs' Principle in cryptography, which holds that a cryptographic system's security should rely solely on the key material.[6] For steganography to remain undetected, the unmodified cover medium must be kept secret, because if it is exposed, a comparison between the cover and stego media immediately reveals the changes.

Information theory allows us to be even more specific on what it means for a system to be perfectly secure. Christian Cachin proposed an information-theoretic model for steganography that considers the security of steganographic systems against passive eavesdroppers.[7] In this model, you assume that the adversary has complete knowledge of the encoding system but does not know the secret key. His or her task is to devise a model for the probability distribution P_C of all possible cover media and P_S of all possible stego media. The adversary can then use *detection theory* to decide between hypothesis C (that a message contains no hidden information) and hypothesis S (that a message carries hidden content). A system is perfectly secure if no decision rule exists that can perform better than random guessing.

Essentially, steganographic communication senders and receivers agree on a steganographic system and a shared secret key that determines how a message is encoded in the cover medium. To send a hidden message, for example, Alice creates a new image with a digital camera. Alice supplies the steganographic system with her shared secret and her message. The steganographic system uses the shared secret to determine how the hidden message should be encoded in the redundant bits. The result is a stego image that Alice sends to Bob. When Bob receives the image, he uses the shared secret and the agreed on steganographic system to retrieve the hidden message. Figure 1 shows an overview of the encoding step; as mentioned earlier, statistical analysis can reveal the presence of hidden content.[8–12]

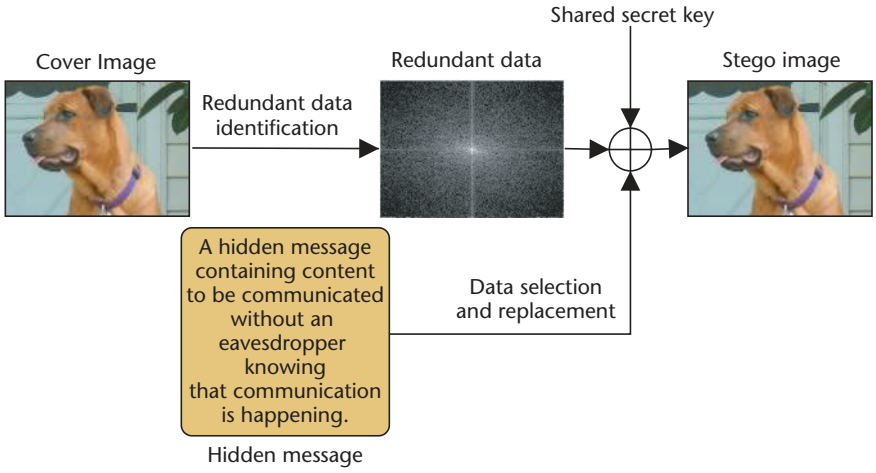

Figure 1. Modern steganographic communication. The encoding step of a steganographic system identifies redundant bits and then replaces a subset of them with data from a secret message.

Hide and seek

Although steganography is applicable to all data objects that contain redundancy, in this article, we consider JPEG images only (although the techniques and methods for steganography and steganalysis that we present here apply to other data formats as well). People often transmit digital pictures over email and other Internet communication, and JPEG is one of the most common formats for images. Moreover, steganographic systems for the JPEG format seem more interesting because the systems operate in a transform space and are not affected by visual attacks.[8] (Visual attacks mean that you can see steganographic messages on the low bit planes of an image because they overwrite visual structures; this usually happens in BMP images.) Neil F. Johnson and Sushil Jajodia, for example, showed that steganographic systems for palette-based images leave easily detected distortions.[9]

Let's look at some representative steganographic systems and see how their encoding algorithms change an image in a detectable way. We'll compare the different systems and contrast their relative effectiveness.

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

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

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

(a)

(b)

Figure 2. Embedded information in a JPEG. (a) The unmodified original picture; (b) the picture with the first chapter of *The Hunting of the Snark* embedded in it.

Input: message, cover image
Output: stego image
while data left to embed **do**
    get next DCT coefficient from cover image
    if DCT ≠ 0 and DCT ≠ 1 **then**
        get next LSB from message
        replace DCT LSB with message LSB
    end if
    insert DCT into stego image
end while

Figure 3. The JSteg algorithm. As it runs, the algorithm sequentially replaces the least-significant bit of discrete cosine transform (DCT) coefficients with message data. It does not require a shared secret.

$$F^{Q}(u,v) = \left\lfloor \frac{F(u,v)}{Q(u,v)} \right\rfloor,$$

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

We can use the least-significant bits of the quantized DCT coefficients as redundant bits in which to embed the hidden message. The modification of a single DCT coefficient affects all 64 image pixels.

In some image formats (such as GIF), an image's visual structure exists to some degree in all the image's bit layers. Steganographic systems that modify least-significant bits of these image formats are often susceptible to visual attacks.[8] This is not true for JPEGs. The modifications are in the frequency domain instead of the spatial domain, so there are no visual attacks against the JPEG image format.

Figure 2 shows two images with a resolution of 640 × 480 in 24-bit color. The uncompressed original image is almost 1.2 Mbytes (the two JPEG images shown are about 0.3 Mbytes). Figure 2a is unmodified; Figure 2b contains the first chapter of Lewis Carroll's *The Hunting of the Snark*. After compression, the chapter is about 15 Kbytes. The human eye cannot detect which image holds steganographic content.

Sequential

Derek Upham's JSteg was the first publicly available steganographic system for JPEG images. Its embedding algorithm sequentially replaces the least-significant bit of DCT coefficients with the message's data (see Figure 3).[13] The algorithm does not require a shared secret; as a result, anyone who knows the steganographic system can retrieve the message hidden by JSteg.

Andreas Westfeld and Andreas Pfitzmann noticed that steganographic systems that change least-significant bits sequentially cause distortions detectable by steganalysis.[8] They observed that for a given image, the embedding of high-entropy data (often due to encryption) changed the histogram of color frequencies in a predictable way.

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.

The same is true in the JPEG data format. Instead of measuring color frequencies, we observe differences in the DCT coefficients' frequency. Figure 4 displays the histogram before and after a hidden message is embedded in a JPEG image. We see a reduction in the frequency difference between coefficient –1 and its adjacent DCT coefficient –2. We can see a similar reduction in frequency difference between coefficients 2 and 3.

Westfeld and Pfitzmann used a χ^2-test to determine whether the observed frequency distribution y_i in an image matches a distribution y_i^* that shows distortion from embedding hidden data. Although we do not know the cover image, we know that the sum of adjacent DCT coefficients remains invariant, which lets us compute the expected distribution y_i^* from the stego image. Letting n_i be the DCT histogram, we compute the arithmetic mean

$$y_i^* = \frac{n_{2i} + n_{2i+1}}{2}$$

to determine the expected distribution and compare it against the observed distribution

$$y_i = n_{2i}.$$

The χ^2 value for the difference between the distributions is given as

$$\chi^2 = \sum_{i=1}^{v+1} \frac{\left(y_i - y_i^*\right)}{y_i^*},$$

where v are the degrees of freedom—that is, one less than the number of different categories in the histogram. It might be necessary to sum adjacent values from the expected distribution and the observed distribution to ensure that each category has enough counts. Combining two adjacent categories reduces the degrees of freedom by one. The probability p that the two distributions are equal is given by the complement of the cumulative distribution function,

$$p = 1 - \int_0^{\chi^2} \frac{t^{(v-2)/2} e^{-t/2}}{2^{v/2}\Gamma(v/2)},$$

where Γ is the Euler Gamma function.

The probability of embedding is determined by calculating p for a sample from the DCT coefficients. The samples start at the beginning of the image; for each measurement the sample size is increased. Figure 5 shows the probability of embedding for a stego image created by JSteg. The high probability at the beginning of the image reveals the presence of a hidden message; the point at which the probability drops indicates the end of the message.

Pseudo random

OutGuess 0.1 (created by one of us, Niels Provos) is a steganographic system that improves the encoding step by using a pseudo-random number generator to select DCT coefficients at random. The least-significant bit of a selected DCT coefficient is replaced with encrypted message data (see Figure 6).

The χ^2-test for JSteg does not detect data that is randomly distributed across the redundant data and, for that reason, it cannot find steganographic content hidden by OutGuess 0.1. However, it is possible to extend the χ^2-test to be more sensitive to local distortions in an image.

Two identical distributions produce about the same χ^2 values in any part of the distribution. Instead of increasing the sample size and applying the test at a constant position, we use a constant sample size but slide the position where the samples are taken over the image's entire range.

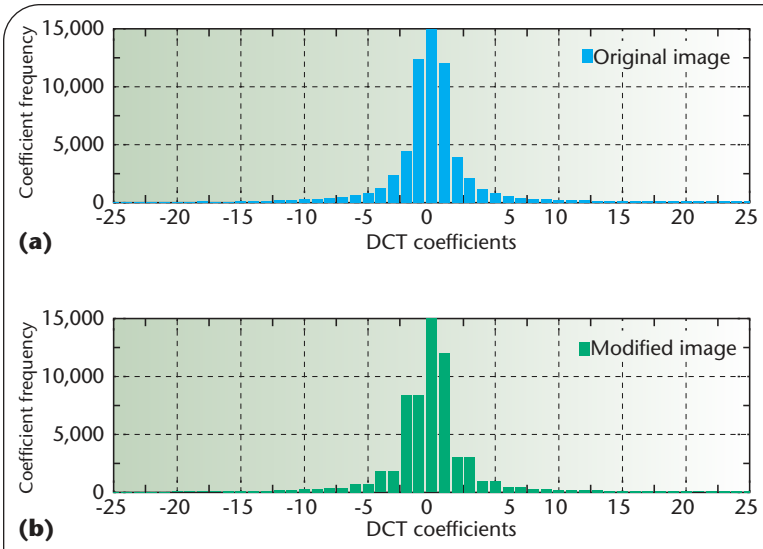

Figure 4. Frequency histograms. Sequential changes to the (a) original and (b) modified image's least-sequential bit of discrete cosine transform coefficients tend to equalize the frequency of adjacent DCT coefficients in the histograms.

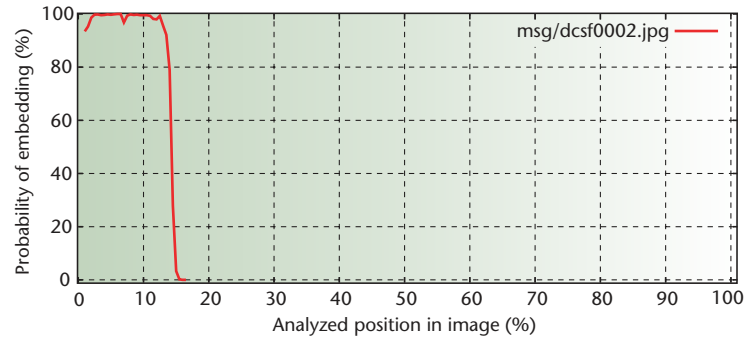

Figure 5. A high probability of embedding indicates that the image contains steganographic content. With JSteg, it is also possible to determine the hidden message's length.

```
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 ≠ 0 and DCT ≠ 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.

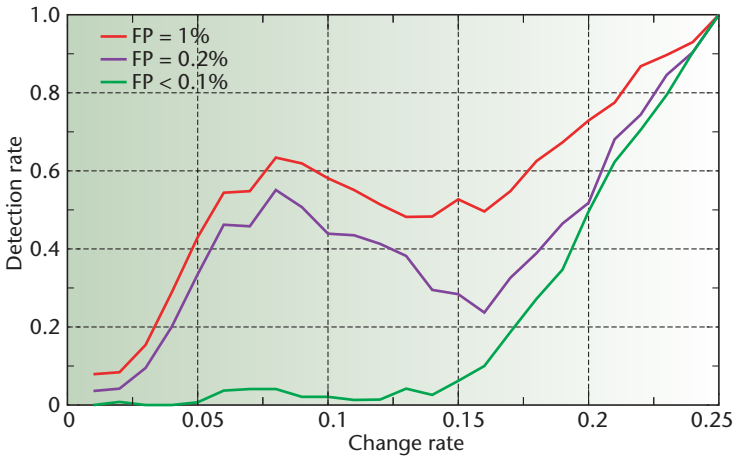

Figure 7. The extended χ^2-test detects pseudo-randomly embedded messages in JPEG images. The detection rate depends on the hidden message's size and can be improved by applying a heuristic that eliminates coefficients likely to lead to false negatives. The graph shows the detection rates for three different 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.

Using the extended test, we can detect pseudo-randomly distributed hidden data.

Given a constant sample size, we take samples at the beginning of the image and increase the sample position by 1 percent for every χ^2 calculation. We then take the sum of the probability of embedding for all samples. If the sum is greater than the detection threshold, the test indicates that an image contains a hidden message.

To find an appropriate sample size, we select an expected distribution for the extended χ^2-test that should cause a negative test result. Instead of calculating the arithmetic mean of coefficients and their adjacent ones, we take the arithmetic mean of two unrelated coefficients,

$$y_i^* = \frac{n_{2i-1} + n_{2i}}{2}.$$

A binary search on the sample size helps find a value for which the extended χ^2-test does not show a correlation to the expected distribution derived from unrelated coefficients.

Figure 7 shows an analysis of the extended χ^2-test for different false-positive rates. Its detection rate depends on the hidden data's size and the number of DCT coefficients in an image. We characterize their respective relation by using the *change rate*—the fraction of DCT coefficients available for embedding a hidden message that have been modified. With a false-positive rate of less than 0.1 percent, the extended χ^2-test starts detecting embedded content for change rates greater than 5 percent. We improve the detection rate by using a heuristic that eliminates coefficients likely to lead to false negatives. Due to the heuristic, the detection rate for embedded content with a change rate of 5 percent is greater than 40 percent for a 1 percent false-positive rate.

One of us (Niels Provos) showed that applying correcting transforms to the embedding step could defeat steganalysis based on the χ^2-test.[12] He observed that not all the redundant bits were used when embedding a hidden message. If the statistical tests used to examine an image for steganographic content are known, it is possible to use the remaining redundant bits to correct statistical deviations that the embedding step created. In this case, preserving the DCT frequency histogram prevents steganalysis via the χ^2-test.

Siwei Lyu and Hany Farid suggested a different approach based on discrimination of two classes: stego image and non-stego image.[10,11] Statistics collected from images in a training set determine a function that discriminates between the two classes. The discrimination function determines the class of a new image that is not part of the training set. The set of statistics used by the discrimination function is called the *feature vector*.

Lyu and his colleague used a *support vector machine* (SVM) to create a nonlinear discrimination function. Here, we present a less sophisticated but easier to understand method for determining a linear discrimination function,

$$\Lambda(\mathbf{X}) = \sum_{i=1}^{k} b_i x_i ,$$

of the measured image statistics $\mathbf{X} = (x_1, x_2, \ldots, x_k)^T$ that, for appropriately chosen b_i, discriminates between the

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

two classes.

For a new image $\mathbf{X}$, the discriminant function Λ lets us decide between two hypotheses: the hypothesis H_0 that the new image contains no steganographic content and the hypothesis H_1 that the new image contains a hidden message.

For the binary hypothesis problem, detection theory provides us with the Neyman-Pearson criterion, which shows that the likelihood ratio test

$$\Lambda(\mathbf{X}) = \frac{p_{\mathbf{x}}|H_1(\mathbf{X}|H_1)}{p_{\mathbf{x}}|H_0(\mathbf{X}|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \eta$$

maximizes the detection rate P_D for a fixed false-negative rate P_F,[14] where $p_{\mathbf{x}}|H_1$ ($\mathbf{X}|H_1$) and $p_{\mathbf{x}}|H_0(\mathbf{X}|H_0)$ denote the joint probability functions for $(x_1, x_2, \ldots, x_k)$ under H_1 and H_0, respectively. The constant η is the detection threshold.

To choose the weights b_i, we assume that the set x_i of non-stego images and the set y_i of stego images are independently and normally distributed. This assumption lets us calculate the probability functions $p_{\mathbf{x}}|H_1(\mathbf{X}|H_1)$ and $p_{\mathbf{x}}|H_0(\mathbf{X}|H_0)$, which we use to derive the weights b_i.

Determining the discrimination functions is straightforward, but finding a good feature vector is difficult. Farid created a feature vector with a wavelet-like decomposition that builds higher-order statistical models of natural images.[10] He derived the statistics by applying separable low- and high-pass filters along the image axes generating vertical, horizontal, and diagonal subbands, which are denoted $V_i(x,y)$, $H_i(x,y)$ and $D_i(x,y)$, respectively, for different scales $i = 1, \ldots, n$.

The first set of statistics for the feature vector is given by the mean, variance, skewness, and kurtosis of the subband coefficients at each orientation and at scales $i = 1, \ldots, n-1$. The second set of statistics is based on the errors in an optimal linear predictor of coefficient magnitude. For each subband and scale, the error's distribution is characterized by its mean, variance, skewness, and kurtosis resulting in a total size of $24(n-1)$ for the feature vector.

Lyu and Farid's training set consists of 1,800 non-stego images and a random subset of 1,800 stego images that contain images as hidden content. Using four different scales, a program (or a researcher) calculates a 72-length feature vector for each image in the training set. Table 1 shows their achieved detection rate using a nonlinear SVM for false-positive rates 0.0 percent and 1.0 percent and different message sizes.

The discrimination function works well only if the training set captures the image space's useful characteristics. For different types of images—for example, nature scenes and indoor photographs—the detection rate could decrease when using a single training set. Improving the training set by selecting images that match the type of image we're analyzing might be possible. The probability models for clutter in natural images that Ulf Grenander and Anuj Srivastava[15] first proposed let us select similar images from the training set automatically.

We can improve the detection quality rate by using a feature vector based on different statistics. Instead of using a wavelet-like decomposition, we look at the distribution of squared differences,

$$H_{ik} = \frac{\sqrt{\sum_{j=0}^{6}\left(F_i(j+1,k) - F_i(j,k)\right)^2}}{1 + \sum_{j=0}^{6}\left|F_i(j,k)\right|}$$

$$V_{ik} = \frac{\sqrt{\sum_{j=0}^{6}\left(F_i(k,j+1) - F_i(k,j)\right)^2}}{1 + \sum_{j=0}^{6}\left|F_i(k,j)\right|},$$

where i enumerates the number of blocks in the image, and k enumerates the rows or columns in a single block. For each distribution, we calculate the mean and its first three central moments, resulting in 64 measurements for

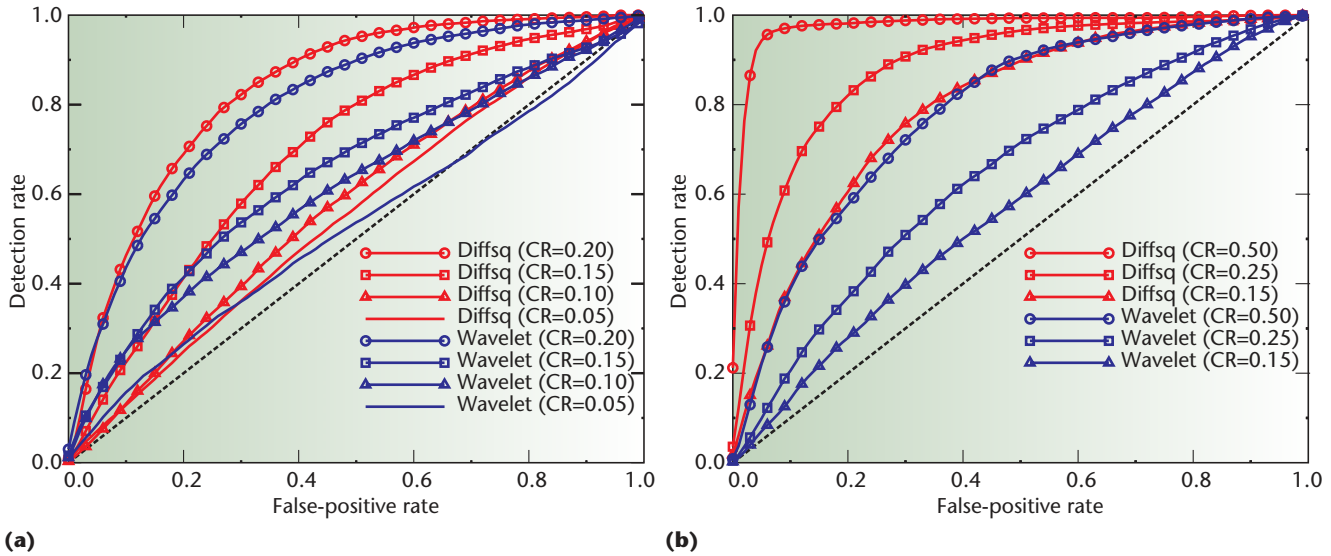

Figure 8. Different feature vectors based on wavelet-like decomposition and on squared differences. (a) The receiver operating characteristic (ROC) for OutGuess detection and (b) the ROC for F5 detection.

a single image.

Figure 8 compares the linear discrimination functions derived from the two feature vectors. Figure 8a shows receiver-operating characteristics (ROC) for OutGuess messages and their corresponding change rates; Figure 8b shows the ROCs for F5 messages (described in more detail later). For OutGuess, the feature vectors show comparable detection performance. However, for F5, the squared differences feature vector outperforms the wavelet feature vector.

Using a discrimination function does not help us determine a hidden message's length. Jessica Fridrich and her colleagues made a steganalytic attack on OutGuess that can determine a hidden message's length.[16] OutGuess preserves the first-order statistics of the DCT coefficients, so Fridrich and her group devised a steganalytic method independent of the DCT histogram. They used discontinuities along the boundaries of 8×8 pixel blocks as a macroscopic quantity that increases with the hidden message's length. The discontinues are measured by the *blockiness* formula

$$B = \sum_{i=1}^{\left\lfloor \frac{M-1}{8} \right\rfloor} \sum_{j=1}^{N} \left| g_{8i,j} - g_{8i+1,j} \right| + \sum_{i=1}^{M} \sum_{j=1}^{\left\lfloor \frac{N-1}{8} \right\rfloor} \left| g_{i,8j} - g_{i,8j+1} \right|,$$

where g_{ij} are pixel values in an $M \times N$ grayscale image.

Experimental evidence shows that the blockiness B increases monotonically with the number of flipped least-sequential bits in the DCT coefficients. Its first derivative decreases with the hidden message's length, meaning that the blockiness function's slope is maximal for the cover image and decreases for an image that already contains a message.

Using the blockiness measure, the algorithm to detect OutGuess proceeds as follows:

1. Determine the blockiness $B_S(0)$ of the decompressed stego image.
2. Using OutGuess, embed a maximal length message and calculate the resulting stego image's blockiness $B_S(1)$.
3. Crop the stego image by four pixels to reconstruct an image similar to the cover image. Compress the resulting image using the same JPEG quantization matrix as the stego image and calculate the blockiness $B(0)$.
4. Using OutGuess, embed a maximal length message into the cropped image and calculate the resulting blockiness $B(1)$.
5. Using OutGuess, embed a maximal length message into the stego image from the previous step and compute the resulting blockiness $B1(1)$.
6. The slope $S_0 = B(1) - B(0)$ corresponds to the original cover image, and $S_1 = B1(1) - (1)$ is the slope for an image with an embedded, maximal length message. The stego image's slope $S = B_S(1) - B_S(0)$ is between the two slopes S_0 and S_1. The hidden message's length is then determined as

$$p = \frac{S_0 - S}{S_0 - S_1},$$

where $p = 0$ corresponds to the cover image and $p = 1$ to an image with the maximal embedded message length.

To counter randomness in the OutGuess embedding algorithm, we repeat the detection algorithm 10 times. The average of the p-values is taken as the final message length.

Fridrich and her group tested their algorithm on 70 images of which 24 contained hidden messages. Their analysis showed an error in the estimated message length of –0.48 percent ± 6 percent. This approach has two advantages over class discrimination: it does not require a training set and it determines the length of hidden messages.

Subtraction

Steganalysis successfully detects steganographic systems that replace the least-significant bits of DCT coefficients. Let's turn now to Andreas Westfeld's steganographic system, F5.[17]

Instead of replacing the least-significant bit of a DCT coefficient with message data, F5 decrements its absolute value in a process called *matrix encoding*. As a result, there is no coupling of any fixed pair of DCT coefficients, meaning the χ^2-test cannot detect F5.

Matrix encoding computes an appropriate $(1, (2^k - 1), k)$ Hamming code by calculating the message block size k from the message length and the number of nonzero non-DC coefficients. The Hamming code $(1, 2^k - 1, k)$ encodes a k-bit message word m into an n-bit code word a with $n = 2^k - 1$. It can recover from a single bit error in the code word.[18]

F5 uses the decoding function $f(a) = \oplus_{i=1}^{n} a_i \cdot i$ and the Hamming distance d. With matrix encoding, embedding any k-bit message into any n-bit code word changing it at most by one bit. In other words, we can find a suitable code word a' for every code word a and every message word m so that $m = f(a')$ and $d(a, a') \leq 1$. Given a code word a and message word m, we calculate the difference $s = m \oplus f(a)$ and get the new code word as

$$a' = \begin{cases} a & \text{if } s = 0 \\ (a_1, a_2, \ldots, \neg a_s, \ldots, a_n) & \text{otherwise} \end{cases}$$

Figure 9 shows the F5 embedding algorithm. First, the DCT coefficients are permuted by a keyed pseudo-random number generator (PRNG), then arranged into groups of n while skipping zero and DC coefficients. The message is split into k-bit blocks. For every message block m, we get an n-bit code word a by concatenating the least-significant bit of the current coefficients' absolute value. If the message block m and the decoding $f(a)$ are the same, the message block can be embedded without any changes; otherwise, we use $s = m \oplus f(a)$ to determine which coefficient needs to change (its absolute value is decremented by one). If the coefficient becomes zero, *shrinkage* happens, and it is discarded from the coefficient group. The group is filled with the next nonzero coefficient and the process repeats until the message can be embedded.

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

For smaller messages, matrix encoding lets F5 reduce the number of changes to the image—for example, for $k = 3$, every change embeds 3.43 message bits while the total code size more than doubles. Because F5 decrements DCT coefficients, the sum of adjacent coefficients is no longer invariant, and the χ^2 test cannot detect F5-embedded messages.

However, Fridrich and her group presented a steganalytic method that does detect images with F5 content.[19] They estimated the cover-image histogram from the stego image and compared statistics from the estimated histogram against the actual histogram. As a result, they found it possible to get a modification rate β that indicates if F5 modified an image.

Fridrich and her colleagues' steganalysis determined how F5's embedding step changes the cover image's AC coefficients. Let

$$h_{uv}(d) := |\{F(u,v) \mid d = |F(u,v)|, u + v \neq 0\}|$$

be the total number of AC DCT coefficients in the cover image with frequency (u,v) whose absolute value equals

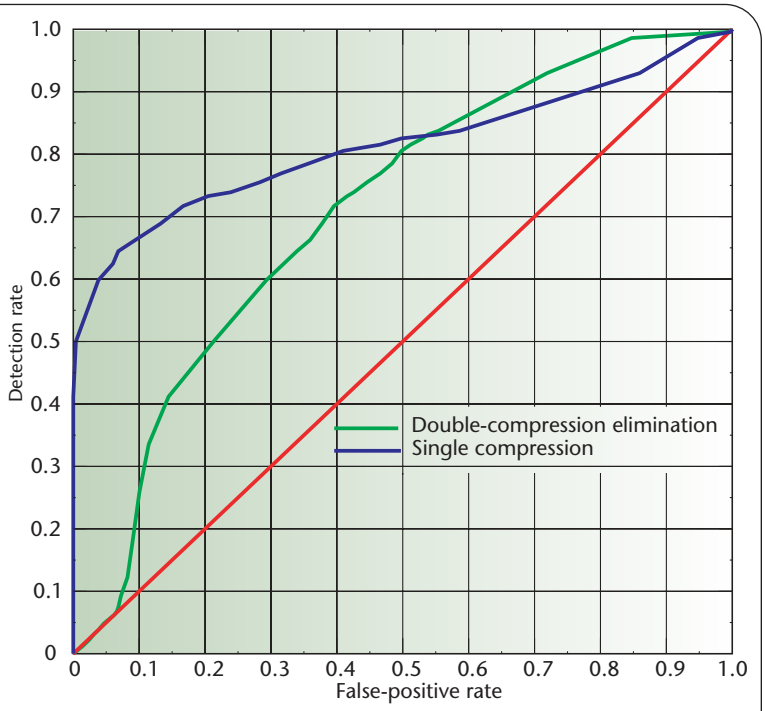

Figure 10. Receiver-operating characteristics (ROCs) of the F5 detection algorithm. The detection rate is analyzed when using double compression elimination and against single compressed images.

d. $H_{uv}(d)$ is the corresponding function for the stego image.

If F5 changes n AC coefficients, the change rate β is n/P, where P is the total number of AC coefficients. As F5 changes coefficients pseudo randomly, we expect the histogram values for the stego image to be

$$H_{uv}(d) < (1-\beta)h_{uv}(d) + \beta h_{uv}(d+1), \quad \text{for } d > 0$$

$$H_{uv}(0) < h_{uv}(0) + \beta h_{uv}(1), \quad \text{for } d = 0.$$

Fridrich and her group used this estimate to calculate the expected change rate β from the cover image histogram. They found the best correspondence when using $d = 0$ and $d = 1$ because these coefficient values change the most during the embedding step. This leads to the approximation

$$\beta_{uv} \approx \frac{h_{uv}(1)\left[H_{uv}(1) - h_{uv}(1)\right] + \left[H_{uv}(1) - h_{uv}(1)\right]\left[h_{uv}(2) - h_{uv}(1)\right]}{h_{uv}^2(1) + \left[h_{uv}(2) - h_{uv}(1)\right]^2}.$$

The final value of β is calculated as the average of β_{uv} for the frequencies $(u,v) \in \{(1,2),(2,1),(2,2)\}$.

The histogram values for the cover image are unknown and must be estimated from the stego image. We do this by decompressing the stego image into the spatial domain. The resulting image is then cropped by four pixels on each side to move the errors at the block boundaries. We recompress the cropped image using the same quantization tables as the stego image, getting the estimates for the cover image histogram from the recompressed image.

Because many images are stored already in the JPEG format, embedding information with F5 leads to double compression, which could confuse this detection algorithm. Fridrich and her group proposed a method for eliminating the effects of double compression by estimating the quality factor used to compress the cover image. Unfortunately, they based their evaluation of the detection algorithm on only 20 images. To get a better understanding of its accuracy, we present an evaluation of the algorithm based on our own implementation.

Figure 10 shows the ROC for a test set of 500 non-stego and 500 stego images. In the first test, both types of images are double-compressed due to F5. The only difference is that the stego images contain a steganographic message. Notice that the false-positive rate is fairly high compared to the detection rate. The second test uses the original JPEG images without double compression as reference.

Statistics-aware embedding

So far, we have presented embedding algorithms that overwrite image data without directly considering the distortions that the embedding caused. Let's look at a framework for an embedding algorithm that uses global image statistics to influence how coefficients should be changed.

To embed a single bit, we can either increment or decrement a DCT coefficient's value. This lets us change a DCT coefficient's least-significant bit in two different ways. Additionally, we create groups of DCT coefficients and use the parity[1] of their least-significant bits as message bits to further increase the number of ways to embed a single bit. For every DCT block, we search the space of all possible changes to find a configuration that minimizes the change to image statistics. Currently, we search for solutions that maintain the blockiness, the block variance, and the coefficient histogram.

We are still in the progress of evaluating this approach's effectiveness. However, in contrast to previously presented steganographic systems, the changes our algorithm introduces depend on image properties and take statistics directly into consideration.

Comparison

Detecting sequential changes in the least-significant bits of DCT coefficients (as seen in JSteg) is easy. A simple χ^2-test helps us determine a hidden message's presence and size. Detecting other systems is more difficult, but all the

systems presented here predictably change the cover medium's statistical properties.

Steganographic systems use different methods to reduce changes to the cover medium. OutGuess, for example, carefully selects a special seed for its PRNG; F5 uses matrix encoding. We can also compress the hidden message before embedding it, but even though this reduces the number of changes to the cover medium, the steganographic systems' statistical distortions remain unchanged. For detection algorithms that can determine the hidden message's length, the detection threshold increases only slightly.

We discussed two different classes of detection algorithms: one based on *inherent statistical properties* and the other on *class discrimination*. Detection algorithms based on inherent statistical properties have the advantage that they do not need to find a representative training set; moreover, they often let us estimate an embedded message's length. However, each steganographic system requires its own detection algorithm. Class discrimination, on the other hand, is universal—even though it doesn't provide an estimate of the hidden message's length, and creating a representative training set is often difficult. A feature vector can help detect several steganographic systems, once we get a good training set. It remains to be seen if new steganographic systems can circumvent detection using class discrimination.

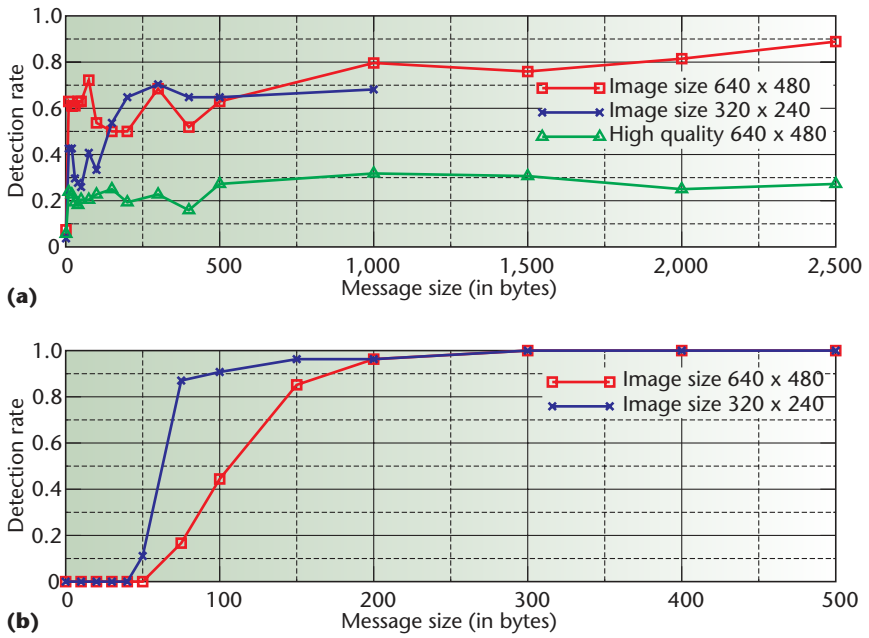

Figure 11. Using Stegdetect over the Internet. (a) JPHide and (b) JSteg produce different detection results for different test images and message sizes.

Steganography detection on the Internet

How can we use these steganalytic methods in a real-world setting—for example, to assess claims that steganographic content is regularly posted to the Internet?[20–22] To find out if such claims are true, we created a steganography detection framework[23] that gets JPEG images off the Internet and uses steganalysis to identify subsets of the images likely to contain steganographic content.

Steganographic systems in use

To test our framework on the Internet, we started by searching the Web and Usenet for three popular steganographic systems that can hide information in JPEG images: JSteg (and JSteg-Shell), JPHide, and OutGuess. All these systems use some form of least-significant bit embedding and are detectable with statistical analysis.

JSteg-Shell is a Windows user interface to JSteg first developed by John Korejwa. It supports content encryption and compression before JSteg embeds the data. JSteg-Shell uses the RC4 stream cipher for encryption (but the RC4 key space is restricted to 40 bits).

JPHide is a steganographic system Allan Latham first developed that uses Blowfish as a PRNG.[24,25] Version 0.5 (there's also a version 0.3) supports additional compression of the hidden message, so it uses slightly different headers to store embedding information. Before the content is embedded, the content is Blowfish-encrypted with a user-supplied pass phrase.

Detection framework

Stegdetect is an automated utility that can analyze JPEG images that have content hidden with JSteg, JPHide, and OutGuess 0.13b. Stegdetect's output lists the steganographic systems it finds in each image or writes "negative" if it couldn't detect any.

We calibrated Stegdetect's detection sensitivity against a set of 500 non-stego images (of different sizes) and stego images (from different steganographic systems). On a 1,200-MHz Pentium III processor, Stegdetect can keep up with a Web crawler on a 10 MBit/s network.

Stegdetect's false-negative rate depends on the steganographic system and the embedded message's size. The smaller the message, the harder it is to detect by statistical means. Stegdetect is very reliable in finding images that have content embedded with JSteg. For JPHide, detection depends also on the size and the compression quality of the JPEG images. Furthermore, JPHide 0.5 reduces

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

the hidden message size by employing compression. Figure 11 shows the results of detecting JPHide and JSteg.

For JSteg, we cannot detect messages smaller than 50 bytes. The false-negative rate in such cases is almost 100 percent. However, once the message size is larger than 150 bytes, our false-negative rate is less than 10 percent. For JPHide, the detection rate is independent of the message size, and the false-negative rate is at least 20 percent in all cases. Although the false-negative rate for OutGuess is around 60 percent, a high false-negative rate is preferable to a high false-positive rate, as we explain later.

Finding images

To exercise our ability to test for steganographic content automatically, we needed images that might contain hidden messages. We picked images from eBay auctions (due to various news reports)[20,21] and discussion groups in the Usenet archive for analysis.[26]

To get images from eBay auctions, a Web crawler that could find JPEG images was the obvious choice. Unfortunately, there were no open-source, image-capable Web crawlers available when we started our research. To get around this problem, we developed Crawl, a simple, efficient Web crawler that makes a local copy of any JPEG images it encounters on a Web page. Crawl performs a depth-first search and has two key features:

- Images and Web pages can be matched against regular expressions; a match can be used to include or exclude Web pages in the search.
- Minimum and maximum image size can be specified, which lets us exclude images that are too small to contain hidden messages. We restricted our search to images larger than 20 Kbytes but smaller than 400.

We downloaded more than two million images linked to eBay auctions. To automate detection, Crawl uses *stdout* to report successfully retrieved images to Stegdetect. After processing the two million images with Stegdetect, we found that over 1 percent of all images seemed to contain hidden content. JPHide was detected most often (see Table 2).

We augmented our study by analyzing an additional one million images from a Usenet archive. Most of these are likely to be false-positives. Stefan Axelsson applied the *base-rate fallacy* to intrusion detection systems and showed that a high percentage of false positives had a significant effect on such a system's efficiency.[27] The situation is very similar for Stegdetect.

We can calculate the true-positive rate—the probability that an image detected by Stegdetect really has steganographic content—as follows

$$P(S|D) = \frac{P(S)\cdot P(D|S)}{P(D)} = \frac{P(S)\cdot P(D|S)}{P(S)\cdot P(D|S) + P(\neg S)\cdot P(D|\neg S)},$$

where $P(S)$ is the probability of steganographic content in images, and $P(\neg S)$ is its complement. $P(D|S)$ is the probability that we'll detect an image that has steganographic content, and $P(D|\neg S)$ is the false-positive rate. Conversely, $P(\neg D|S) = 1 - P(D|S)$ is the false-negative rate.

To improve the true-positive rate, we must increase the numerator or decrease the denominator. For a given detection system, increasing the detection rate is not possible without increasing the false-positive rate and vice versa. We assume that $P(S)$—the probability that an image contains steganographic content—is extremely low compared to $P(\neg S)$, the probability that an image contains no hidden message. As a result, the false-positive rate $P(D|\neg S)$ is the dominating term in the equation; reducing it is thus the best way to increase the true-positive rate. Given these assumptions, the false-positive rate also dominates the computational costs to verifying hidden content. For a detection system to be practical, keeping the false-positive rate as low as possible is important.

Verifying hidden content

The statistical tests we used to find steganographic content in images indicate nothing more than a likelihood that content is embedded. Because of that, Stegdetect cannot guarantee a hidden message's existence.

To verify that the detected images have hidden content, Stegbreak must launch a *dictionary attack* against the JPEG files. JSteg-Shell, JPHide, or Outguess all hide content based on a user-supplied password, so an attacker can try to guess the password by taking a large dictionary and trying to use every single word in it to retrieve the hidden message. In addition to message data, the three systems also embed header information, so attackers can verify a guessed password using header information such as message length. For a dictionary attack[28] to work, the steganographic system's user must select a weak password (one from a small subset of the full password space).

Ultimate success, though, depends on the dictionary's quality. For the eBay images, we used a dictionary with roughly 850,000 words from several languages. For the Usenet images, we improved the dictionary by including

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

four-digit PIN numbers and short pass phrases. We created these short pass phrases by taking three- to five-letter words from a list of the 2,000 most common English words and concatenating them. The resulting dictionary contains 1.8 million words.

We measured Stegbreak's performance on a 1,200-MHz Pentium III by running a dictionary attack against one image and then against a set of 50 images (see Table 3). The speed improvement for 50 images is due to key schedule caching. For JPHide, we checked about 8,700 words per second; a test run with 300 images and a dictionary of roughly 577,000 words took 10 days to check for both versions of JPHide. Blowfish is designed to make key schedule computation expensive, which slowed down Stegbreak. When checking for JPHide 0.5, the Blowfish key schedule must be recomputed for almost every image.

Stegbreak was faster for OutGuess—about 34,000 words per second. However, due to limited header information, a large dictionary can produce many candidate passwords. For JSteg-Shell, Stegbreak checked about 47,000 words per second, which was fast enough to run a dictionary attack on a single computer. JSteg-Shell restricts the key space to 40 bits, but if passwords consist of only 7-bit characters, the effective key space is reduced to 35 bits. We could search that key space in about eight days.

Distributed dictionary attack

Stegbreak is too slow to run a dictionary attack against JPHide on a single computer. Because a dictionary attack is inherently parallel, distributing it to many workstations is possible. To distribute Stegbreak jobs and data sets, we developed Disconcert, a distributed computing framework for loosely coupled workstations.

There are two natural ways to parallelize a dictionary attack: each node is assigned its own set of images or each node is assigned its own part of the dictionary. With more words existing than images, the latter approach permits finer segmentation of the work. To run the dictionary attack, Disconcert hands out work units to workstations in the form of an index into the dictionary. After a node completes a work unit, it receives a new index to work on.

To analyze the eBay images, Stegbreak ran on about 60 nodes at the University of Michigan, 10 of them at the Center for Information Technology Integration. The combined performance required for analyzing JPHide was about 200,000 words per second, 16 times faster than a 1,200-MHz Pentium III. The slowest client contributed 471 words per second to the job; the fastest, 12,504 words per second. For the Usenet images, we increased the cluster's size to 230 nodes. Peak performance was 870,000 keys per second, the equivalent of 72 1,200-MHz Pentium III machines.

For the more than two million images Crawl downloaded from eBay auctions, Stegdetect indicated that about 17,000 seemed to have steganographic content. We observed a similar detection rate for the one million images that we obtained from the Usenet archives. To verify correct behavior of participating clients, we inserted tracer images into every Stegbreak job. As expected, the dictionary attack found the correct passwords for these images.

From our eBay and Usenet research, we so far have not found a single hidden message. We offer four explanations for our inability to find steganographic content on the Internet:

- All steganographic system users carefully choose passwords that are not susceptible to dictionary attacks.
- Maybe images from sources we did not analyze carry steganographic content.
- Nobody uses steganographic systems that we could find.
- All messages are too small for our analysis to detect.

All these explanations are valid to some degree. Yet, even if the majority of passwords used to hide content were strong, we would expect to find weak passwords: one study found nearly 25 percent of all passwords were vulnerable to dictionary attack.[29] Similarly, even if many of the steganographic systems used to hide messages were undetectable by our methods, we would expect to find messages hidden with the popular and accessible systems for JPEG images that are big enough to be detected. That leaves two remaining explanations: either we are looking in the wrong place or there is no widespread use of steganography on the Internet. We are currently researching new algorithms to hide information and also improve steganalysis. □

Acknowledgments

We thank Patrick McDaniel, Bruce Fields, Olga Kornievskaia, José Nazario, and Thérése Pasquesi for careful reviews, Hany Farid and Jessica Fridrich for helpful comments and suggestions, Mark Giuffrida and David Andersen for computing resources, and The Internet Archive *for access to their USENET archives.*

References

1. R.J. Anderson and F.A.P. Petitcolas, "On the Limits of Steganography," *J. Selected Areas in Comm.*, vol. 16, no. 4, 1998, pp. 474–481.
2. F.A.P. Petitcolas, R.J. Anderson, and M.G. Kuhn, "Information Hiding—A Survey," *Proc. IEEE*, vol. 87, no. 7, 1999, pp. 1062–1078.
3. J. Fridrich and M. Goljan, "Practical Steganalysis—State of the Art," *Proc. SPIE Photonics Imaging 2002, Security and Watermarking of Multimedia Contents*, vol. 4675, SPIE Press, 2002, pp. 1–13.
4. B. Chen and G.W. Wornell, "Quantization Index Modulation: A Class of Provably Good Methods for Digital Watermarking and Information Embedding," *IEEE Trans. Information Theory*, vol. 47, no. 4, 2001, pp. 1423–1443.
5. N.F. Johnson and S. Jajodia, "Exploring Steganography: Seeing the Unseen," *Computer*, vol. 31, no. 2, 1998, pp. 26–34.
6. A. Kerckhoffs, "La Cryptographie Militaire (Military Cryptography)," *J. Sciences Militaires (J. Military Science,* in French*)*, Feb. 1883.
7. C. Cachin, *An Information-Theoretic Model for Steganography*, Cryptology ePrint Archive, Report 2000/028, 2002, www.zurich.ibm.com/~cca/papers/stego.pdf.
8. A. Westfeld and A. Pfitzmann, "Attacks on Steganographic Systems," *Proc. Information Hiding—3rd Int'l Workshop*, Springer Verlag, 1999, pp. 61–76.
9. N.F. Johnson and S. Jajodia, "Steganalysis of Images Created Using Current Steganographic Software," *Proc. 2nd Int'l Workshop in Information Hiding*, Springer-Verlag, 1998, pp. 273–289.
10. H. Farid, "Detecting Hidden Messages Using Higher-Order Statistical Models," *Proc. Int'l Conf. Image Processing*, IEEE Press, 2002.
11. S. Lyu and H. Farid, "Detecting Hidden Messages Using Higher-Order Statistics and Support Vector Machines," *Proc. 5th Int'l Workshop on Information Hiding*, Springer-Verlag, 2002.
12. N. Provos, "Defending Against Statistical Steganalysis," *Proc. 10th Usenix Security Symp.*, Usenix Assoc., 2001, pp. 323–335.
13. T. Zhang and X. Ping, "A Fast and Effective Steganalytic Technique Against JSteg-like Algorithms," *Proc. 8th ACM Symp. Applied Computing*, ACM Press, 2003.
14. H.L.V. Trees, *Detection, Estimation, and Modulation Theory, Part I: Detection, Estimation, and Linear Modulation Theory*, Wiley Interscience, 2001.
15. U. Grenander and A. Srivastave, "Probability Models for Clutter in Natural Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 4, 2001.
16. J. Fridrich, M. Goljan, and D. Hogea, "Attacking the OutGuess," *Proc. ACM Workshop Multimedia and Security 2002*, ACM Press, 2002.
17. A. Westfeld, "F5—A Steganographic Algorithm: High Capacity Despite Better Steganalysis," *Proc. 4th Int'l Workshop Information Hiding*, Springer-Verlag, 2001, pp. 289–302.
18. J.H. van Lint, *Introduction to Coding Theory*, 2nd ed. Springer-Verlag, 1992.
19. J. Fridrich, M. Goljan, and D. Hogea, "Steganalysis of JPEG Images: Breaking the F5 Algorithm," *Proc. 5th Int'l Workshop Information Hiding*, Springer-Verlag, 2002.
20. J. Kelley, "Terror Groups Hide Behind Web Encryption," *USA Today*, Feb. 2001, www.usatoday.com/life/cyber/tech/2001-02-05-binladen.htm.
21. D. McCullagh, "Secret Messages Come in .Wavs," *Wired News*, Feb. 2001, www.wired.com/news/politics/0,1283,41861,00.html.
22. J. Kelley, "Militants Wire Web with Links to Jihad," *USA Today*, July 2002, www.usatoday.com/news/world/2002/07/10/web-terror-cover.htm.
23. N. Provos and P. Honeyman, "Detecting Steganographic Content on the Internet," *Proc. 2002 Network and Distributed System Security Symp.*, Internet Soc., 2002.
24. B. Schneier, "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)," *Fast Software Encryption, Cambridge Security Workshop Proc.*, Springer-Verlag, 1993, pp. 191–204.
25. A. Latham, "Steganography: JPHIDE and JPSEEK," 1999; http://linux01.gwdg.de/~alatham/stego.html.
26. "The Internet Archive: Building an 'Internet Library'," 2001; www.archive.org.
27. S. Axelsson, "The Base-Rate Fallacy and its Implications for the Difficulty of Intrusion Detection," *Proc. 6th ACM Conf. Computer and Comm. Security*, ACM Press, 1999, pp. 1–7.
28. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.
29. D. Klein, "Foiling the Cracker: A Survey of, and Improvements to, Password Security," *Proc. 2nd Usenix Security Workshop*, Usenix Assoc., 1990, pp. 5–14.

Niels Provos *is an experimental computer scientist conducting research in steganography and in computer and network security. He is a PhD candidate at the University of Michigan and an active contributor to open-source projects. Contact him at provos@citi.umich.edu.*

Peter Honeyman *is scientific director of the Center for Information Technology Integration and adjunct professor of electrical engineering and computer science at the University of Michigan. He is secretary of the Usenix Association, co-vice chair of IFIP WG 8.8, and a member of IFIP WG 6.1, AAAS, and EFF. Contact him at honey@citi.umich.edu.*