Forensic Steganalysis: Determining the Stego Key in Spatial Domain Steganography

[a]Jessica Fridrich*, [a]Miroslav Goljan, [b]David Soukal, and [a]Taras Holotyak

[a]Dept. of Electrical and Computer Engineering, [b]Dept. of Computer Science
SUNY Binghamton, Binghamton, NY 13902-6000, USA

ABSTRACT

This paper is an extension of our work[1] on stego key search for JPEG images published at EI SPIE in 2004. We provide a more general theoretical description of the methodology, apply our approach to the spatial domain, and add a method that determines the stego key from multiple images. We show that in the spatial domain the stego key search can be made significantly more efficient by working with the noise component of the image obtained using a denoising filter. The technique is tested on the LSB embedding paradigm and on a special case of embedding by noise adding (the ±1 embedding). The stego key search can be performed for a wide class of steganographic techniques even for sizes of secret message well below those detectable using known methods. The proposed strategy may prove useful to forensic analysts and law enforcement.

1. INTRODUCTION

The art of discovering secret messages embedded using steganography[2] is called steganalysis. The vast majority of work in steganalysis focuses on detection of secret messages rather than extraction. On a more general level, steganalysis comprises of several phases, some of which belong to digital forensic analysis (hence the term Forensic Steganalysis in the title of this paper): 1) identification of suspicious images, 2) determining the steganographic method in use, 3) searching for the stego key and extracting the embedded bit-stream, 4) deciphering the bit-stream. In this paper, we investigate Phase 3 under the assumption that we have one or more stego images and, by Kerckhoffs' principle, we already know the steganographic program used for embedding (i.e., we have the source code).

One simple approach to determine the stego key would be to use a brute-force search for the stego key, inspecting the most likely keys first (dictionary attack) and extracting the alleged message while looking for a recognizable header as a sign that we have come across the correct stego key[3]. However, this approach will fail if the embedded data stream does not have any detectable structure in which case the search also becomes significantly more complicated because for each stego key, all possible encryption keys must be tested. Thus, the complexity of the brute force search is proportional to the product of the size of stego and encryption keyspaces. Even though for some stego programs the stego key space itself may be small enough to make the brute force search for the stego key plausible, if the message has been encrypted using strong encryption, the search becomes computationally infeasible.

Trivedi *et al.*[4,5] presented a method for secret key detection in sequential steganography. The authors' goal is to determine, using a sequential probability ratio test, the embedding key, which is, in their interpretation, the beginning and the end of the subsequence modulated during embedding. In contrast, in this paper the key determines a *pseudo-randomly ordered subset* of all indices in the cover signal to be used for embedding. This situation is more typical for a steganographic application, while sequential embedding is typically used for watermarking. While it is possible to apply the method of Ref. 4 for this case by performing the same hypothesis test for each possible key, additional research would have to be done to estimate the probability of falsely determined and missed keys. Also, the necessity to encounter a jump in the statistics implies that the whole signal used for embedding must be processed, which would slow down the search.

In this paper, we follow the approach previously proposed for JPEG images[1] and modify it for spatial domain steganography. In the next section, we define the embedding paradigm that will be investigated in this paper and in Section 3 we give a detailed problem formulation. The stego key search method itself is described in Section 4. In

* fridrich@binghamton.edu; phone 1 607 777-2577; fax 1 607 777-4464; http://www.ws.binghamton.edu/fridrich

Section 5, experimental results are interpreted and discussed for ±1 embedding and Least Significant Bit embedding (LSB) in the spatial domain. In Section 6, we show how the reliability of the search can be improved if multiple stego images embedded with the same key are available. Finally, the paper is concluded in Section 7 where we discuss limitations of the proposed method and possible countermeasures.

2. THE EMBEDDING PARADIGM

The stego-key search method described in this paper is applicable to virtually all steganographic methods whose embedding mechanism consists of the following three primitives in sequence: 1) The embedding proceeds along a pseudo-random path generated from the stego key, 2) The message bits are embedded as parities of individual pixels (e.g. their LSBs, special palette parity assignments[6], key-dependent parities[7], etc.), 3) If necessary, the pixels' parity is changed using an embedding operation. Indeed, most steganographic techniques work in this manner. The random path selection is usually implemented using a Pseudo-Random Number Generator (PRNG) that is seeded with a seed derived from a user-specified stego key or a pass-phrase. The output of the PRNG is used to generate a pseudo-random walk through the pixels.

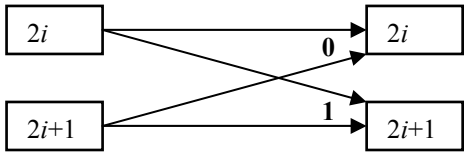


Figure 1a. LSB embedding operation

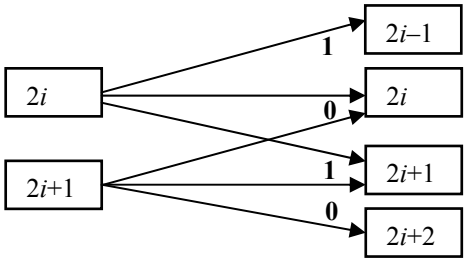


Figure 1b. ±1 embedding operation

The secret message is embedded in the form of a bit-stream as the parity of the elements along the pseudo-random walk. In order to match the element parity with the message bit, the element is modified using an embedding operation. This operation flips the element parity and it could be deterministic or probabilistic. For LSB embedding, the element parity is defined as its LSB and the embedding operation is the LSB flipping. Fig. 1a shows how even and odd element values $2i$ and $2i+1$ are potentially modified to embed a specific message bit during LSB embedding.

The program Hide[8] also hides data in LSBs of pixels (e.g., it is based on the same parity mapping), however, it uses a different probabilistic embedding operation (see Fig. 1b). In other words, when the message bit does not match the pixel parity (its LSB), Hide either adds or subtracts 1 with equal probability with the exception of values 0 and 255, which are only increased or decreased, respectively.

3. PROBLEM FORMULATION

One possible approach to the stego key search would be to first identify which elements have been modified and then try to reverse-engineer the PRNG that generated the embedding path. However, this approach is infeasible because it is in general very hard to identify which elements have been modified. Second, even if we were able to identify the modified elements, we will not know the order in which they were modified and we will not know the complete path because, on average, 50% of elements were not modified as their parity already matched the message bit. Third, most PRNG are very hard to reverse-engineer in the sense of identifying the seed from the PRN sequence. For a cryptographically strong PRNG, this task is as complex as an exhaustive search for the key. Consequently, it seems that the only way to find the stego key is to use an exhaustive search, possibly combined with the dictionary attack. The key search algorithm proposed in this paper is of this type, as well.

To avoid any confusion, we formulate the stego key search more precisely. The steganographic algorithm may employ some form of a many-to-one mapping (e.g., a hash function) to map the user pass-phrase (or password) to the seed for the PRNG. As a result, it may not be feasible to identify the user password itself using any search method. Thus, in this paper, when we speak about searching for the stego key, we are in fact *searching for the seed that was used to initialize the PRNG rather than the user pass-phrase itself*. In other words, if our search is successful, we will be able to find the correct seed, the embedding path, and extract the embedded bits even though we may not be able to recover the pass-phrase. Having said this, for convenience we will nevertheless speak about stego key search to mean that we are, in fact, searching for the PRNG seed.

Throughout this text, boldface symbols will denote vectors or matrices, non-boldface symbols will stand for scalars, and boldface Greek symbols will denote random variables. We reserve the letter i to index image elements, k for indices of histogram bins, and j to index stego keys.

Let the cover image be represented with a vector $\boldsymbol{x}=\{x_i\}$, $i=1, \ldots, N$. Depending on the image format, the elements x_i can be shades of gray or color indices. The range of x_i is a finite set of integers. Let K be the space of all possible stego keys that lead to different pseudo-random paths. After embedding m message bits, the stego image, $\boldsymbol{x}(q) = \{x_i(q)\}$, is obtained, where $q = m/N$ is the relative message length. During embedding, at least $m \le N$ elements in $\boldsymbol{x}$ are visited (and potentially modified) along the path generated from the stego key $K_0 \in \mathsf{K}$. Our task is to find the embedding stego key K_0 given only the stego image $\boldsymbol{x}(q)$ and a full knowledge of the embedding algorithm.

For each possible candidate key $K_j \in \mathsf{K}$, let $Path(K_j)$ denote the ordered set of element indices visited along the path generated from the key K_j. Assuming the embedded message bits are i.i.d. realizations of a binary random variable uniformly distributed on $\{0, 1\}$ (which is the case if the message is encrypted), in the sequence $\{x_i(q)\}$, $i \in Path(K_0)$, on average 50% of elements were modified by the embedding operation. Thus, taking the first n elements along the path generated from the correct key, $n \le m$, the expected number of modified elements is $n/2$.

Assuming paths produced from different keys are independent and that each path of length n forms a random subset of the cover image (each element has the same probability of being selected), the probability of encountering a modified element is $m/(2N)$. Thus, the expected number of modified elements along an *incorrect* path consisting of n elements is $n \times m/(2N) \le n/2$ (because $m \le N$). Thus, if the stego image is not fully embedded, the distribution of elements $\{x_i(q)\}$, $i \in Path(K_0)$, along the correct path will be different from the distributions taken along the incorrect paths $\{x_i(q)\}$, $i \in Path(K_j)$, $j > 0$. Assuming the elements $x_i(q)$ are i.i.d. realizations of a random variable, the elements' Probability Density Function (PDF) is their complete statistical characterization. Thus, we identify the correct key as the one for which the distribution of elements $x_i(q)$ along the embedding path is not compatible with the PDF derived for incorrect keys.

As explained in Section 4, it is possible to calculate from the whole stego image $\boldsymbol{x}(q)$ the expected distribution $\boldsymbol{h}$ of image elements $\{x_i(q)\}$ along paths generated from an *incorrect* key. Thus, the stego key search involves a composite hypothesis testing for each candidate key K_j:

H_0: the elements $\{x_i(q)\}$, $i \in Path(K_j)$, are drawn from $\boldsymbol{h}$.
H_1: the elements $\{x_i(q)\}$, $i \in Path(K_j)$, are not drawn from $\boldsymbol{h}$.

For this purpose, we use the chi-square test. One of the reasons for this choice is the low computational complexity of this test, which is crucial for any exhaustive search method. Keys for which the null hypothesis is rejected are possible candidates for the correct key and are further inspected (see Section 4.1).

4. STEGO KEY SEARCH USING THE CHI-SQUARE TEST

In order to apply the chi-square test, we divide the range of elements $x_i(q)$, $i=1, \ldots, N$, into d disjoint bins $B_1, B_2, \ldots, B_d$. The choice of bins depends on the steganographic technique and is discussed in detail in Section 5. The discrete distribution of the first n elements along the path generated from key K_j will be denoted using $h_k(K_j, n, q)$, $k = 1, \ldots, d$. In other words, nh_k is the number of elements among the first n elements $x_i(q)$, $i \in Path(K_j)$, whose values belong to the k–th bin B_k. Note that $h_k(K_j, n, 0)$ is the same quantity calculated from the *cover* image $\boldsymbol{x}$. Furthermore, let $h_k(q)$, $k = 1, \ldots, d$, denote the distribution of *all* image elements from the whole image $\boldsymbol{x}(q)$.

Let Ξ denote the random variable that stands for a randomly selected *incorrect* key from K (each key selected with the same probability). The random variable $h_k(\Xi, n, q)$ has a multivariate hypergeometric distribution with the expected value and variance of $h_k(\Xi, n, q)$ (for proof, see for example Ref. 9):

$$E\{h_k(\Xi, n, q)\} = h_k(q) \tag{1}$$

$$Var\{h_k(\Xi, n, q)\} = \frac{1}{n} h_k(q)(1 - h_k(q)) \frac{N-n}{N-1}. \tag{2}$$

For $n < 0.05N$, $h_k(\Xi, n, q)$ is well approximated using multivariate binomial distribution. If, at the same time, n is large enough to warrant that each bin is sufficiently populated (at least 30 samples in each bin[9]), then the binomial distribution is well approximated with a Gaussian distribution. These conditions will be satisfied in practice, because for digital images N is typically of the order of millions, while n is at most of the order of thousands (also, see the discussion for choosing the bins in Section 5). Therefore, with $n, N \to \infty$ and $n < 0.05N$, the variable S

$$S(\Xi, n, q) = n\frac{N-1}{N-n}\sum_{k=1}^{d}\frac{\left(h_k(\Xi, n, q) - h_k(q)\right)^2}{h_k(q)} \tag{3}$$

is asymptotically chi-square distributed with d–1 degrees of freedom.

We now calculate the value of the statistic S for the *correct* key K_0

$$\begin{aligned} S(K_0, n, q) &= n\frac{N-1}{N-n}\sum_{k=1}^{d}\frac{\left(h_k(K_0, n, q) - h_k(q)\right)^2}{h_k(q)} \\ &= n\frac{N-1}{N-n}\sum_{k=1}^{d}\frac{\left(h_k(K_0, n, q) - h_k(1)\right)^2 + \left(h_k(1) - h_k(q)\right)^2 + 2\left(h_k(K_0, n, q) - h_k(1)\right)\left(h_k(1) - h_k(q)\right)}{h_k(q)}. \end{aligned} \tag{4}$$

The dominant term in the numerator is the middle term $(h_k(1)–h_k(q))^2$. This is because along the correct path, the values $x_i(q)$, $i \in Path(K_0)$, follow the same distribution as elements randomly chosen from a fully embedded image. Thus, $h_k(K_0, n, q)$ can be considered as a sample mean drawn from N realizations of a random variable ζ_k with probability distribution $\text{Prob}(\zeta_k=1) = h_k(1)$, $\text{Prob}(\zeta_k=0) = 1–h_k(1)$. The expected value and variance of the sample mean is[9] $h_k(1)$ and $\frac{1}{n}h_k(1)\left(1-h_k(1)\right)\frac{N-n}{N-1}$, respectively. Consequently, the first and third terms in (4) vanish with increasing n while the second term is non-zero and independent of n. Therefore, for the correct key K_0

$$S(K_0, n, q) \approx n\frac{N-1}{N-n}\sum_{k=1}^{d}\frac{\left(h_k(1) - h_k(q)\right)^2}{h_k(q)}. \tag{5}$$

So far, in our considerations, the embedded message was a fixed random binary bit-stream – qN realizations of an i.i.d. binary random variable uniformly distributed on $\{0,1\}$. Realizing the messages as a qN-dimensional vector binary random variable $\boldsymbol{\mu}$ uniformly distributed in $\{0,1\}^{qN}$, $\boldsymbol{h}(q)$ becomes a k-dimensional vector random variable that we denote $\boldsymbol{h}(\boldsymbol{\mu}, q)$. For a large class of steganographic schemes, there is a linear relationship between $\boldsymbol{h}(0)$ (the histogram of elements of the cover image) and the expected value of $\boldsymbol{h}(\boldsymbol{\mu}, q)$

$$E\left\{h_k(\boldsymbol{\mu}, q)\right\} = \sum_{l=1}^{d}(A_{kl}q + C_{kl})h_l(0), \tag{6}$$

where $\boldsymbol{A}$ and $\boldsymbol{C}$ are constant $d \times d$ matrices. For long messages, $E\{h_k(\boldsymbol{\mu}, q)\} \approx h_k(q)$, which simplifies (5) to

$$S(K_0, n, q) \approx n\frac{N-1}{N-n}(1-q)^2\sum_{k=1}^{d}\frac{1}{h_k(q)}\left(\sum_{l=1}^{d}A_{kl}h_l(0)\right)^2. \tag{7}$$

Assuming that all bins in the histogram of elements of the embedded image are populated, e.g., $h_k(q) \geq 1/N$ for all $q \in [0,1]$, we see that $\rho(q) = \sum_{k=1}^{d}\frac{1}{h_k(q)}\left(\sum_{l=1}^{d}A_{kl}h_l(0)\right)^2$ is a bounded function of q on $[0,1]$. Thus, $S(K_0, n, q)$ decreases to zero as $(1–q)^2$ when q approaches 1. This confirms the intuition that the key search should become less reliable for messages whose length approaches the maximal image capacity.

The linear relationship (6) is satisfied for many steganographic schemes. In particular, it is true for *any* steganography that can be formulated as adding noise that is independent of the cover image element values because then $E\{\boldsymbol{h}(\boldsymbol{\mu}, q)\}$ is a convolution of $\boldsymbol{h}(0)$ with a low-pass filter kernel[11].

The performance of the key search will be measured using the probability

$$p(n,q) = \mathrm{Prob}\big(S(\Xi,n,q) \geq S(K_0,n,q)\big) \approx \frac{\left(\frac{1}{2}S(K_0,n,q)\right)^{\frac{d-3}{2}} e^{-\frac{1}{2}S(K_0,n,q)}}{\Gamma\left(\frac{d-1}{2}\right)} \tag{8}$$

that during the stego key search a randomly chosen incorrect key will produce a value of the statistic S equal or larger than the value obtained for the correct key (7). Expression (8) is obtained using the asymptotic expansion of the cumulative density function F_{d-1} (c.d.f.) for the chi-square distribution with d–1 degrees of freedom (which is an incomplete Gamma function):

$$1-F_{d-1}(x) = \mathrm{Prob}(S(\Xi,n,q) \geq x) = \frac{1}{2^{\frac{d-1}{2}}\Gamma\left(\frac{d-1}{2}\right)}\int_x^\infty e^{-\frac{t}{2}} t^{\frac{d-1}{2}-1} dt = \frac{1}{\Gamma\left(\frac{d-1}{2}\right)}\left(\frac{x}{2}\right)^{\frac{d-3}{2}} e^{-\frac{x}{2}}\left(1+O\left(\frac{1}{x}\right)\right) . \tag{9}$$

The expected number of incorrect outlier keys K_j producing $S(K_j, n, q) \geq S(K_0, n, q)$ among N_K keys is

$$N_{\mathrm{out}} = N_K\, p(n, q). \tag{10}$$

Note that the chi-square value for the correct key (7) increases with n. Thus, larger values of n will lead to a smaller number of candidate keys (10) at the expense of more computations. Also, n needs to be large enough so that our assumption about (3) being asymptotically chi-square distributed is satisfied. Obviously, we also need to keep n smaller than the number of embedded bits, $n < m = qN$. If q can be estimated using quantitative steganalysis methods[10], we can use this estimate and choose n accordingly. If q cannot be estimated, it is in our interest to keep n small to be able to detect stego keys for short messages and to maximize the search speed. Typically, $n \sim 500$–10000 provides a good compromise between the above mentioned requirements.

Also, note from (7) and (10) that the number of outliers N_{out} gradually increases as q approaches 1 (see Fig. 3). This will slow down the key search as more candidate keys must be further inspected using complement checking or other measures (Section 4.1).

4.1 Search speed and candidates for the correct key

Because the size of the key space varies significantly among steganographic systems and can be quite large, an essential property of an effective stego key search algorithm is its speed with which it processes individual keys. To maximize the processing speed *and* the probability of finding the correct key in a reasonable amount of time, one can employ several measures:

a) The stego key search should start with a dictionary attack and inspect the most likely keys first.
b) The number of image elements n along each path could be varied for each key based on the evidence we collect as we add more elements[12].
c) The testing may consist of several hierarchical passes. All keys are first processed using a fast detector with an extremely low probability of missing a correct key but possibly with a high false positive rate. This will produce a smaller set of keys that is further processed using another test that has higher reliability but also higher computational complexity. We can cascade several detectors in this manner to maximize the speed of the search algorithm.
d) For many steganographic techniques, it is possible to estimate[11] the relative message length q. This estimate gives us information on how to choose n and how many false outliers N_{out} can be expected during the search.

It is possible that more than one key pass Step c) above. In fact, the number of keys that are identified as potentially correct is given by (10) and strongly depends on the relative message length $q=m/N$, the number of image elements n, the properties of the cover image $\rho(q)$, and the number of inspected keys N_K. To identify the correct key, for each candidate key we can determine the whole embedding path and inspect n image elements that were not visited dur-

ing embedding and were thus unmodified (*complement checking*). For an incorrect key, we expect statistical evidence compatible with an incorrect key (e.g., a low value of S), while for the correct key the elements' distribution should again produce an outlier value of S.

Another possibility to identify the correct key from outliers is to gradually increase n while looking for a "sudden" change in the statistic S as we encounter the end of the message (c.f., Westfeld's "chi-square attack"[13]). However, this approach requires always $O(N)$ operations for every incorrect key, which increases with image size and thus slows down the key search.

Finally, we note that one of the most important factors influencing the speed of the key search is the PRNG used for generating the random paths. Steganographic algorithms that generate a random permutation of *all* image elements before embedding will lead to slower key searches than algorithms for which only a small portion of each path can be generated without having to produce the whole embedding path (e.g., OutGuess). In fact, deliberately making the path generation slow, e.g., one second, can be considered as a countermeasure against key search as it will slow down any exhaustive searches for key.

5. STEGO KEY SEARCH IN SPATIAL DOMAIN

The search algorithm as described above is directly applicable only to images in the JPEG format. For steganographic systems that work in the spatial domain, before applying this methodology, the stego image should be preprocessed in the following manner. We apply a denoising filter F to the stego image and calculate the residual $\boldsymbol{r}(q) = \boldsymbol{x}(q) - F(\boldsymbol{x}(q))$ with elements $r_i(q)$. We have experimented with simple FIR filters, the Wiener filter, and some nonlinear filters. The best performance was obtained using a wavelet-based denoising filter (Appendix A). The filtering improves the SNR between the stego signal and the cover image. It also decorrelates the stego image elements. Thus, our assumption to model the image elements as an i.i.d. signal becomes more plausible. This preliminary step improves the performance of the stego key search quite dramatically.

In this paper, we address two major embedding types – LSB embedding and ±1 embedding (Fig. 1), which are the simplest examples of embedding by noise adding[7]. We have chosen LSB embedding because most steganographic schemes available on the Internet use this simple embedding paradigm. The ±1 embedding was chosen as an example of a scheme for which no detection is currently known that would work for a wide class of images.

For our testing, we used a "generic" Matlab implementation of the LSB and ±1 embedding in which the secret key is used as a seed for a PRNG. The output of the PRNG is used to spread the message bits at pseudo-random positions in the stego image. To speed up our simulations, we used a special fast random-path generator that enables generation of the first n image elements without having to generate the complete embedding path.

For LSB embedding, we further pre-process the image elements utilizing the fact that we know the pixel modifications are in LSBs only. We calculate the residual $\boldsymbol{r}(q) = \boldsymbol{x}(q) - F(\boldsymbol{x}(q))$ with elements $r_i(q)$, and the "shifted" residual $\bar{\boldsymbol{r}}(q) = \bar{\boldsymbol{x}}(q) - F(\boldsymbol{x}(q))$, with elements $\bar{r}_i(q)$, where $\bar{\boldsymbol{x}}(q)$ denotes $\boldsymbol{x}(q)$ with all its LSBs flipped. Because along an incorrect path, fewer pixels are modified than along the correct path, the average value of $r_i(q)$ along the correct path is larger than along an incorrect path. On the other hand, the average value of $\bar{r}_i(q)$ along the correct path is smaller than along an incorrect path. Thus, it makes sense to use the difference between the residual and the shifted residual $\bar{r}_i(q) - r_i(q)$ for the chi-square test. Indeed, this significantly improved the search performance in our tests. In the next two paragraphs, we discuss the choice of the bins B_i for the chi-square test.

For LSB embedding, the values $\bar{r}_i(q) - r_i(q)$, $i = 1, \ldots, N$, are divided into bins $B_1, \ldots, B_d$ in the following manner. The bins' width is equal to $\sigma_{\bar{\boldsymbol{r}}-\boldsymbol{r}}/\alpha$, where $\sigma_{\bar{\boldsymbol{r}}-\boldsymbol{r}}$ is the standard deviation of $\bar{r}_i(q) - r_i(q)$, α is a constant, and the bins are evenly distributed around zero. The left most and right most bins are exceptions, spanning to $-\infty$ and $+\infty$, respectively. We observed similar performance for values in the range $0.8 \le \alpha \le 1.1$, $7 \le d \le 10$, and used $\alpha = 0.9$, $d = 8$ in all our tests for LSB. Because for natural images both $\boldsymbol{r}$ and $\bar{\boldsymbol{r}}$ have approximately Gaussian distribution, this choice of bins also guarantees that all bins will be well populated for our analysis of Section 4 to apply.

The choice of bins for the ±1 embedding was different. Because $\boldsymbol{r}(q)$ is approximately zero-mean and has a symmetrical PDF, we can reduce the number of operations in the chi-square test by taking the absolute value of the

residual $|\boldsymbol{r}(q)|$ with all bins in the interval $[0,+\infty)$. The bins' width was again chosen as σ_r/α with the same value of $\alpha = 0.9$ and with $d = 5$.

We have performed a number of different experiments in order to gain understanding of which factors influence the key search the most. As the first simple experiment, we searched for the correct key among 2^{20} keys in one image embedded with ±1 embedding (see Fig. 2).

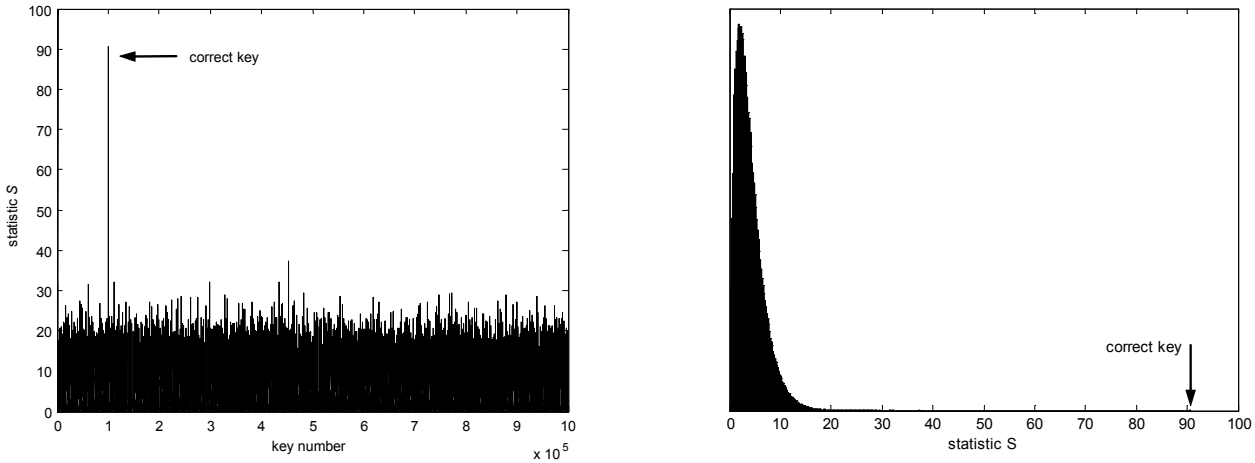


Figure 2. Statistic S (3) (left) and its PDF (right) generated from 2^{20} keys K_j

The performance of the search is quite understandably sensitive to the amount of noise in the image. We took four grayscale images of one scene using the Canon G4 digital camera (image Gazebo in Appendix B) – one image in the raw (uncompressed) format, and three decompressed JPEG images with three different quality settings. The performance of the key search was measured using the probability $p(n, q)$ (8). As can be seen from Table 1, the stego key search works best for the lowest quality (decompressed) JPEG image and worst for the raw image. This is not surprising because the JPEG compression removes high frequency noise and thus the denoising filter F gives a better estimate of the cover image. We can see that for LSB embedding, the stego key search works significantly better overall than for ±1 embedding. The search can also be carried out faster because fewer elements n need to be processed to determine the correct key.

Next, we studied how the stego key search depends on the image content. We experimented with grayscale images of natural scenes containing both indoor and outdoor scenes taken under varying light conditions, all obtained with the Olympus 3030 digital camera, resampled from 2048×1536 pixels to 800×600 pixels, and saved in the 8-bit grayscale format. For illustration, we show $p(n, 0.2)$ for 12 images in Table 2.

The performance of the key search is very strongly influenced by image content, namely its noise component. Image No. 5 has an extreme amount of edges and a strong noise level due to low light conditions. As a result, the key search cannot be successfully completed with a relatively small n. For this image and the LSB method, the smallest n to achieve $p(n, 0.2) \leq 10^{-10}$ is $n \approx 29000$ or 6% of the image size. On the other hand, Image No. 4 has very little structure and the stego key search works extremely reliably. The test images No. 4 and 5 are shown in Appendix B.

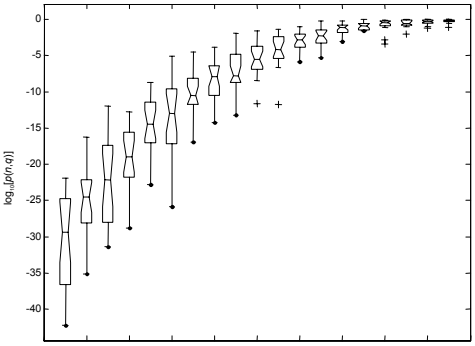
	high compression		medium		low compression		raw image	
n	LSB	±1	LSB	±1	LSB	±1	LSB	±1
5000	–99.02	–26.12	–91.71	–22.60	–53.39	–11.94	–26.42	–1.04
10000	–203.67	–54.96	–160.88	–45.71	–113.15	–25.78	–55.79	–7.45
15000	–317.48	–97.66	–254.05	–67.39	–184.68	–47.05	–81.51	–11.09
20000	–430.66	–128.70	–360.50	–100.34	–249.13	–60.25	–106.71	–16.05

Table 1. Quantity $\log_{10}[p(n,0.2)]$ for 4 image qualities (image Gazebo) averaged over 10 different embeddings

Fig. 3 shows the outlier probability $p(n, q)$ for different relative message length q averaged over 20 different embeddings for each q. One can clearly see how the outlier probability increases as q approaches 1 thus slowing down the key search (as discussed in Section 4). We also see that for short messages, $p(n, q)$ exhibits quite a large variance over different embeddings.

	LSB			±1		
Image #	n=10000	n=20000	n=40000	n=10000	n=20000	n=40000
1	–111.70	–224.05	–531.51	–19.87	–37.09	–95.19
2	–61.39	–158.74	–317.81	–12.22	–29.64	–60.54
3	–18.31	–40.86	–110.32	–1.13	–3.78	–20.40
4	–224.54	–492.18	–1104.06	–43.86	–101.61	–243.26
5	–2.94	–5.93	–14.64	–0.11	–0.20	–0.21
6	–114.09	–254.82	–549.07	–48.47	–128.41	–279.93
7	–165.11	–363.03	–773.72	–63.78	–133.15	–318.05
8	–129.43	–256.11	–519.33	–35.62	–76.39	–153.53
9	–85.89	–202.93	–491.85	–18.14	–57.04	–143.07
10	–125.30	–297.02	–654.55	–31.97	–74.83	–162.58
11	–58.34	–125.33	–270.96	–7.56	–20.17	–51.69
12	–69.20	–140.93	–266.50	–9.97	–21.05	–38.82

Table 2. Quantity $\log_{10}[p(n,0.2)]$ averaged over 10 different embeddings.



ngs for image No. 8 with n=8000 elements. The boxes indicate lower quartile, median, and upper quartile values.

We now address the complexity of the key search. Note that the filtering as well as binning of each value r_i can be done only once before the key search begins. For each candidate key K_j, we need to generate the first n elements of $Path(K_j)$ (time needed is $t(n)$), then construct the histogram of their bin indices (n additions), and finally calculate the statistic (3) ($3d$ arithmetic operations). Note that $t(n)$ is at least linear in n and could even be proportional to N, the number of all pixels, for some steganographic programs for which the complete embedding path must be generated. Thus, $t(n)$ is usually the dominant term for the key search complexity.

We have implemented a generic LSB embedder with a path generator that produced the whole embedding path ($t(n)=O(N)$) to extract the first n elements. In particular, we used the C++ Standard Template Library function std::random_shuffle, a 800×600 grayscale image, $n = 0.05N$, and a Pentium IV machine HT (hyper threading) running at 2.4GHz, 512MB, 3200 DDR RAM. The necessity to generate the whole embedding path slowed down the key search considerably, producing only 11 keys/second.

6. STEGO KEY SEARCH USING MULTIPLE IMAGES

In the case when the stego image is of low quality (noisy) or contains a complex texture or when the key space is very large, our key search algorithm may not provide enough evidence about the correct secret key (there may be too many candidate keys) even after applying the measures of Section 4.1. It is not unreasonable to assume, however, that a forensic analyst will have more than one stego image embedded with the same stego key, which increases her chances to identify the correct key. Let us assume that the analyst has u stego images $J_1, \ldots, J_u$ embedded with the same key K_0, but possibly different messages. Although the measure $p(n, q)$ (8) may not provide convincing evidence about the correct key for each particular image (see Table 3 that shows $p(n, q)$ for the correct key and one incorrect key), some cumulative evidence obtained from all images may uniquely and decisively determine

the correct key. It is not clear, however, how such evidence should be calculated and how the performance of the stego key search should be measured for multiple images.

For each image J_i and key K_j, let $\alpha_i(K_j) = 1 - F_{d-1}\left(S(K_j, n, q)\right)$, where S is defined in (3). Recalling (8), the performance of the key search for the single image J_i was evaluated using

$$\begin{aligned} p_i(n, q) &= \text{Prob}\left(S(\Xi, n, q) \geq S(K_0, n, q)\right) \\ &= \text{Prob}\left(F_{d-1}\left(S(\Xi, n, q)\right) \geq F_{d-1}\left(S(K_0, n, q)\right)\right) = \text{Prob}\left(\alpha_i\left(\Xi\right) < \alpha_i\left(K_0\right)\right), \end{aligned}$$

which is the probability that a randomly chosen key will produce a value of S larger than the one for the correct key K_0 for image J_i. As a (heuristic) cumulative evidence for key K_j from u images, we take the product $\alpha(K_j) = \alpha_1(K_j)\ldots\alpha_u(K_j)$. Obviously, the smaller $\alpha(K_j)$ is, the larger our evidence for the key K_j. Generalizing (8) to u images and dropping the dependence on n and q for brevity, we define

$$p(u) = \text{Prob}\left(\alpha(\Xi) < \alpha(K_0)\right) \tag{11}$$

as the measure of performance for the stego key search for u images.

	J_1	J_2	J_3	J_4	$\alpha(K_j)$	$p(u)$
K_0: $p(n, q)$	1.58×10^{-4}	1.26×10^{-3}	2.00×10^{-12}	3.97×10^{-5}	1.58×10^{-23}	4.04×10^{-19}
K_1: $p(n, q)$	7.16×10^{-5}	8.03×10^{-4}	6.39×10^{-2}	2.21×10^{-1}	8.12×10^{-10}	1.44×10^{-6}

Table 3. Example of collecting evidence from 4 images. Note that while the evidence in favor of each key is inconclusive, the cumulative measure $p(u)$ allows reaching an unambiguous decision when all four images are considered at the same time.

The expected number of incorrect keys (outliers) that produce values $\alpha(K_j) < \alpha(K_0)$ is $N_{\text{out}}(u) = N_K p(u)$. To calculate $p(u)$, we apply Theorem 1 below (proved in Appendix C) to the case when $F_i = F_{d-1}$, $X_i = S(\Xi, n, q)$, for image J_i, and $q(X_1, \ldots, X_u) = \prod_{i=1}^{u}\left(1 - F_{d-1}\left(X_i\right)\right) = \alpha_1(\Xi)\ldots\alpha_u(\Xi)$. Thus, from (11) and (13) we have

$$N_{\text{out}}(u) = N_K p(u) = N_K \alpha(K_0) \sum_{i=0}^{u-1} \frac{(-\log \alpha(K_0))^i}{i!}. \tag{12}$$

Theorem 1. *Let* $q(x_1, x_2, \ldots, x_u) = \prod_{i=1}^{u}(1 - F_i(x_i))$ *be a function of u real variables* x_i*, where* F_i *are cumulative density functions of u independent variables* X_i*. If* F_i^{-1} *exists for all* X_i *(i.e.,* $F_i^{-1}\left(F_i(x)\right) = x$ *for all x and i), then for* $0 < \alpha \leq 1$

$$\text{Prob}(q(X_1, \ldots, X_u) < \alpha) = \frac{\Gamma(u, -\log\alpha)}{(u-1)!} = \alpha\left(1 + \frac{-\log\alpha}{1!} + \frac{(-\log\alpha)^2}{2!} + \ldots + \frac{(-\log\alpha)^{u-1}}{(u-1)!}\right). \tag{13}$$

7. SUMMARY AND COUNTERMEASURES

In this paper, we present a methodology for identifying the stego key from one or more stego images embedded using a key-dependent steganographic scheme. This work is thus a bridge between steganography detection and message extraction and is likely to be of interest to law enforcement and forensic analysts.

In our approach, we focus on steganographic techniques in the spatial domain that embed one bit per image element. We assume that we have a complete knowledge of the embedding algorithm and at least one stego image. The stego key search does not rely on any recognizable patterns in the embedded bit-stream (i.e., it can be encrypted). Instead, the stego key is determined through an exhaustive stego key search by an outlier value of an appropriately defined statistic that quantifies statistical properties of pixels along portions of the embedding path.

We derive expressions for the expected number of falsely determined keys as a function of the relative embedded message length, the image content, and the number of pixels taken along each tested path. We discuss measures that can be taken to identify the correct key from possible candidate keys determined by the search.

Although the search methodology is applicable to virtually all steganographic schemes, this paper focuses on two different embedding paradigms – the LSB and ±1 embedding in the spatial domain. For spatial steganography, prior to the stego key search a special non-linear high-pass filter is applied to the stego image to improve the SNR between the stego signal and the cover image residual. The denoising filter has a major effect on the search performance. By designing a filter matched to each steganographic embedding mechanism, the stego key search performance can likely be further improved.

The existence of fast stego key search algorithms underlines the need for strong steganographic keys. Combining a strong encryption algorithm with an insufficient stego key space may actually lead to successful attacks on the embedding scheme. If the stego key search can be searched in a reasonable time, this method could be used as a detection method.

Besides making the stego keyspace large or slowing down the pseudo-random path generator, there is one simple countermeasure that effectively prevents stego key searches similar to the one described in this paper. If the embedding scheme can, in principle, use every image element with the same probability, independently of the message length, our stego key search will fail. However, padding messages to their maximal length would not be safe as this would make the stego channel more vulnerable to attacks. Instead, we recommend the selection channel[2] or the matrix embedding[14]. For both methods, on average the groups along the correct embedding path will have the same properties as groups along an incorrect path. Moreover, the matrix embedding minimizes the number of embedding changes, which further increases the steganographic security. Lastly, steganographic schemes that use the wet paper codes[15] provide an elegant and effective countermeasure because the sender does not have to share the pixel selection rule with the recipient. Thus, even if the attacker guesses the correct stego key, she will have no information at all about which pixels were used for embedding.

In the case when the stego image is of low quality (noisy) or contains a complex texture or when the keyspace is very large, our key search algorithm may not provide enough evidence about the correct secret key (there may be too many candidate keys) even after applying the measures of Section 4.1. Clearly, having more than one stego image embedded with the same stego key increases our chances to identify the correct key. We proposed a heuristic measure that combines the evidence obtained from multiple images and improves the reliability of the key search.

ACKNOWLEDGEMENTS

The work on this paper was supported by the Air Force Research Laboratory, Air Force Material Command, USAF, under research grant number F30602-02-2-0093. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation there on. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Air Force Research Laboratory, or the U. S. Government.

APPENDIX A

The denoising filter is based on the work first proposed in Ref. 16 and extended in Ref. 17. It is constructed in the wavelet domain while modeling the wavelet coefficients as a conditionally independent zero-mean Gaussian mixture process with identically distributed highly correlated variances. In this appendix, we consider the cover and stego images as two-dimensional signals indexed using two indices, i and j, where i and j are row and column indices. For a message of relative length q, let $x_{ij}(q)$, x_{ij}, and $s_{ij}(q)$ be the values of the stego image, cover image, and the stego signal, respectively. Assuming the act of message embedding is an additive process in the spatial domain, $x_{ij}(q)=x_{ij}+s_{ij}(q)$, we have $X_{ij}(q)=X_{ij}+S_{ij}(q)$ for the corresponding wavelet transforms $X_{ij}=W(x_{ij})$, etc. In this appendix, capital letters will denote wavelet transforms of corresponding lower case variables. We model $S_{ij}(q)$ as a white Gaussian noise $N(0,\sigma_S^2)$, X_{ij} as a locally stationary i.i.d. signal with zero mean, and build the denoising filter in two stages. For justification of the stego message model, see Ref. 18. In the first stage, we estimate the cover image variance

$$\hat{\sigma}^2_{X_{ij}} = \max\left[0, \frac{1}{N_{ij}}\sum_{i,j} X^2_{ij}(q) - \sigma^2_S\right], \tag{A1}$$

where N_{ij} is the number of wavelet coefficients in a local square neighborhood of the (i, j)-th wavelet coefficient and σ_S is calculated from the assumption that the stego image is embedded with a maximal length message (q=1). If the relative message length q can be determined[10], we can also use it to calculate $\sigma_S(q)$. In particular, for both LSB and ±1 embedding $\sigma_S(q) = \sqrt{q/2}$. The second stage uses the local Wiener filter to obtain an estimate of the denoised image $\hat{X}_{ij}$ in the wavelet domain

$$\hat{X}_{ij}(q) = \frac{\hat{\sigma}^2_{X_{ij}}}{\hat{\sigma}^2_{X_{ij}} + \sigma^2_S} X_{ij}(q)\,. \tag{A2}$$

The denoised stego image $F(x_{ij}(q))$ is obtained as the inverse wavelet transform of $\hat{X}_{ij}: F(x_{ij}(q)) = W^{-1}(\hat{X}_{ij})$. Because of the presence of edges in images, we estimate $\hat{\sigma}^2_{X_{ij}}$ not only in local windows of different sizes, but also of different orientations by changing the window size and shape.

To provide some measure of performance of this filter compared to other standard denoising filters, we calculated the correlation between the stego signal $s_{ij}(q)$ and $x_{ij}(q)$–$F(x_{ij}(q))$. The typical value obtained from the 5×5 Wiener filter was around 0.06, compared to 0.31 with the denoising filter.

APPENDIX B

Image Gazebo (2272×1704)

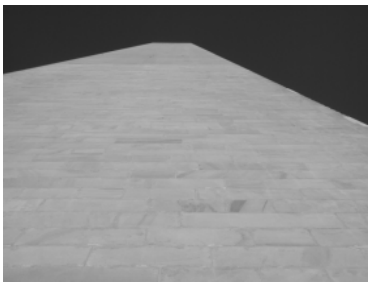


Image No. 4 (1280×960 resampled to 800×600)

Image No. 5 (1280×960 resampled to 800×600)

APPENDIX C

Proof of Theorem 1: The random variables $W_i = \left(1 - F_i(X_i)\right)$, $i = 1,\ldots, u$, are uniformly distributed in [0,1] because $P(W_i < w) = \text{Prob}\left(F_i(X_i) > 1 - w\right) = \text{Prob}\left(X_i > F_i^{-1}(1-w)\right) = 1 - F_i(F_i^{-1}(1-w)) = w$. Because $X_1,\ldots, X_u$ are independent, $Z_u = q(X_1, \ldots, X_u)$ is a product of u independent and uniformly distributed random variables in [0,1] with PDF $f_{Z_u}(z)$,

$$f_{Z_u}(z) = \begin{cases} \dfrac{1}{(u-1)!}(-\log z)^{u-1}, & z \in (0,1) \\ 0, & \text{otherwise,} \end{cases} \tag{C1}$$

which can be proved by induction with respect to u. Thus,

$$\begin{aligned} \text{Prob}(q(X_1,\ldots,X_u) < \alpha) = F_{Z_u}(\alpha) &= \int_{-\infty}^{\alpha} f_{Z_u}(x)\,\mathrm{d}x = \int_0^{\alpha} \frac{1}{(u-1)!}(-\log x)^{u-1}\,\mathrm{d}x \\ &= \frac{1}{(u-1)!}\int_{-\log\alpha}^{\infty} t^{u-1}e^{-t}\,\mathrm{d}t = \frac{\Gamma(u, -\log\alpha)}{(u-1)!} \end{aligned} \tag{C2}$$

where $\Gamma(u, x)$ is the incomplete gamma function. In the final step of the proof, we applied a well-known property of $\Gamma(u, x)$, which can be easily proved by induction

$$\Gamma(u, -\log\alpha) = (u-1)!\alpha\left(1 + \frac{-\log\alpha}{1!} + \frac{(-\log\alpha)^2}{2!} + \cdots + \frac{(-\log\alpha)^{u-1}}{(u-1)!}\right). \quad \text{(C3)}$$

REFERENCES

1. J. Fridrich, M. Goljan, and D. Soukal, "Searching for the Stego Key", *Proc. SPIE Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, San Jose, 2004, pp. 70–82.
2. R. J. Anderson and F. A. P. Petitcolas, "On the Limits of Steganography," *IEEE Journal of Selected Areas in Communications*, Special Issue on Copyright and Privacy Protection, vol. **16**(4), 1998, pp. 474–481.
3. N. Provos and P. Honeyman, "Detecting Steganographic Content on the Internet," CITI Technical Report 01-11, 2001.
4. S. Trivedi and R. Chandramouli, "Locally Most Powerful Detector for Secret Key Estimation in Spread Spectrum Data Hiding," in E. Delp (ed.): *Proc. SPIE*, *Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, San Jose, 2004, pp. 1–12.
5. S. Trivedi and R. Chandramouli, "Secret Key Estimation in Sequential Steganography," to appear in *IEEE Trans. on Signal Processing*, Supplement on Secure Media, February 2005.
6. J. Fridrich and R. Du, "Secure Steganographic Methods for Palette Images," in A. Pfitzmann, (ed.): Information Hiding. 3rd International Workshop. *Lecture Notes in Computer Science*, vol. 1768. Springer-Verlag, Berlin Heidelberg New York, 2000, pp. 47–60.
7. J. Fridrich and M. Goljan, "Digital Image Steganography Using Stochastic Modulation," in E. Delp (ed.): *Proc. SPIE Electronic Imaging*, *Security, Steganography, and Watermarking of Multimedia Contents V*, vol. 5020, Santa Clara, 2003, pp. 191–202.
8. T. Sharp, "An Implementation of Key-Based Digital Signal Steganography," in I. S. Moskowitz, (ed.): Information Hiding. 4th International Workshop, *Lecture Notes in Computer Science*, vol. 2137. Springer-Verlag, Berlin Heidelberg New York, 2001, pp. 13–26.
9. M.R. Spiegel, Schaum's Outline of Theory and Problems of Statistics, McGraw-Hill, New York, 3rd edition, 1961.
10. J. Fridrich, M. Goljan, D. Hogea, and D. Soukal, "Quantitative Steganalysis: Estimating Secret Message Length," *ACM Multimedia Systems Journal*, Special Issue on Multimedia Security, vol. **9**(3), 2003, pp. 288–302.
11. J. J. Harmsen and W. A. Pearlman, "Steganalysis of Additive Noise Modelable Information Hiding," in E. Delp (ed.): *Proc. SPIE Electronic Imaging*, *Security, Steganography, and Watermarking of Multimedia Contents V*, vol. 5020, Santa Clara, 2003, pp. 131–142.
12. R. Chandramouli and N. D. Memon, "On Sequential Watermark Detection," *IEEE Transactions on Signal Processing*, vol. **51**(4), Special Issue on Signal Processing for Data Hiding in Digital Media and Secure Content Delivery, 2003, pp. 1034–1044.
13. A. Westfeld and A. Pfitzmann, "Attacks on Steganographic Systems," in A. Pfitzmann (ed.): 3rd International Workshop. *Lecture Notes in Computer Science*, vol. 1768. Springer-Verlag, Berlin Heidelberg New York, 2000, pp. 61–75.
14. R. Crandall, Some Notes on Steganography, posted on Steganography Mailing List, 1998. http://os.inf.tu-dresden.de/~westfeld/crandall.pdf
15. J. Fridrich, M. Goljan, and D. Soukal, "Perturbed Quantization Steganography Using Wet Paper Codes," in *Proc. ACM Multimedia and Security*, Magdeburg, Germany, Sep. 20–21, 2004, pp. 4–15.
16. S. M. LoPresto, K. Ramchandran, and M. T. Orchard, "Image Coding Based on Mixture Modeling of Wavelet Coefficients and a Fast Estimation-Quantization Framework," *Proc. Data Compression Conf.*, March 1997, pp. 221–230.
17. M. K. Michak, I. Kozintsev, and K. Ramchandran, "Low-Complexity Image Denoising Based on Statistical Modeling of Wavelet Coefficients," *IEEE Signal Processing Letters*, vol. **6**(12), 1999, pp. 300–303.
18. T. Holotyak, J. Fridrich, and D. Soukal, "Stochastic Approach to Secret Message Length Estimation in $\pm k$ Embedding Steganography", to appear in E. Delp (ed.): *Proc. SPIE Electronic Imaging*, *Security, Steganography, and Watermarking of Multimedia Contents VII*, San Jose, 2005.