**CITI Technical Report 01-1**

# Probabilistic Methods for Improving Information Hiding

*Niels Provos*
provos@citi.umich.edu

*ABSTRACT*
This paper presents two new methods to improve the information hiding process in steganography. One method uses probabilistic embedding to minimize modifications to the cover medium. The other method employs error-correcting codes, which allow the embedding process to choose which bits to modify. This decreases the likelihood of being detected. In addition, we can hide multiple data sets in the same cover medium, thus providing plausible deniability.

January 31, 2001

Center for Information Technology Integration
University of Michigan
519 West William Street
Ann Arbor, MI 48103-4943

# Probabilistic Methods for Improving Information Hiding

Niels Provos
*Center for Information Technology Integration*
*University of Michigan*
provos@citi.umich.edu

## 1 Introduction

Steganography is the art and science of hiding the fact that communication is happening. While classical steganographic systems depend on keeping the encoding system secret, modern steganography is detectable only if secret information is known, *e.g.*, a secret key. Because of their invasive nature steganographic systems leave detectable traces within an image's characteristics, *e.g.*, its Fourier signature. This allows an eavesdropper to detect images that have been modified, revealing that secret communication is happening. Although, the secrecy of the information is not degraded, its hidden nature is revealed which defeats the whole purpose of steganography.

In general, the information hiding process extracts redundant bits from the cover medium. Redundant bits are those bits that can be modified without destroying the integrity of the cover medium. The embedding process then selects the bits that will be replaced with data from the hidden message. Then, the redundant bits are replaced in the cover medium to create the stego medium.

This paper presents two new methods to improve the selection process. They are completely independent of the extraction step. The first method selects from a family of pseudo-random number generators [3]. Each pseudo-random number generator results in a different bit selection; the selection that causes the fewest changes to the cover medium is used for the embedding. The second method uses error-correcting codes. Error-correcting codes allow the selector to avoid modifications by introducing artificial errors. The result is a higher flexibility in selecting bits. Both methods can be used together to reduce modifications to the cover medium.

A more processor- and space-intensive alternative is to match existing images against the source data in order to minimize the number of modifications required to express the original message. This reverses the usual steganographic approach of embedding hidden information into a specific medium. Instead, the hidden information is examined and a cover medium will be selected that allows embedding with minimal modifications.

The remainder of this paper is organized as follows. Section 2 describes related work in image steganography. Section 3 introduces the prerequisites necessary for secure steganography. In section 4, we illustrate the extraction process, and introduce new methods to improve the embedding of hidden messages. Section 5 provides a brief analysis of hiding messages in the JPEG [12] image format. Finally, we conclude in section 6.

## 2 Related Work

In the following, we use the terminology established by Pfitzmann et al. [8].

Walton [13] authenticates an image by storing its checksum in the least-significant bits of the image. The bits are distributed uniformly over the image with a pseudo-random number generator. The probabilistic embedding in this paper differs by choosing a seed for the pseudo-random number generator that minimizes the necessary changes to the cover medium.

Aura [1] uses a pseudo-random permutation generator to select the bits in the cover-medium. He notes that if the secret key and cover size remain unchanged, that the selected bits will be the same. In our embedding process, however, the pseudo-random number generator is reseeded to find the best embedding and the bit selection also depends on the hidden message size.

Johnson [4] and Jajiodia have analyzed images created with available steganographic software. They claim that current steganographic techniques leave signatures in the stego medium, but especially for the common JPEG format they fail to show what signatures can be detected. Our implementation does not seem to exhibit the distortion in the discrete cosine transform (DCT) coefficients.

Marvel [5] et al. use Reed-Solomon codes [7] to recover from errors introduced in the transmission process. In our case, we assume error-free transmission and are thus allowed to introduce artificial errors to increase the flexibility of the selection algorithm.

Westfeld and Pfitzmann [14] find visual attacks against common steganographic tools. The only program that embeds data in the JPEG image format, can be detected with a statistical approach because it does not spread the hidden message over all the redundant bits. In our case, however, we spread the hidden message over all the redundant bits.

# 3 Prerequisites

For steganography to remain undetected, the original cover medium needs to be an unknown. Otherwise a comparison between cover medium and stego medium reveals changes. While an adversary gains knowledge of only approximately half of the embedded bits, she still detects a modification.

Zöllner et al. [2] propose an information theoretic approach to solve the problem of secure steganography by employing an nondeterministic selection. In their model, the original medium is known to the adversary but a preprocessing step introduces randomness into the cover data. If the adversary can not obtain the actual cover medium, she can not deduce the embedded message by observing differences between the original and the stego medium. They suggest two necessary conditions for secure steganography:

- The secret key is unknown to the adversary.
- The adversary does not know the actual cover medium.

In practice, these two conditions are easily met. It suffices to create a cover medium with a digital camera or by scanning holiday pictures, as long as the unmodified original is not made publicly available. Furthermore, it is plausible to send holiday images as correspondence over the Internet.

# 4 Construction

This section explains the steganographic embedding process in general and introduce new methods to improve on it. We enumerate the different aspects required for an efficient implementation that uses these methods.

## 4.1 Modularity

The task of embedding hidden information into a cover medium can be divided into two steps:

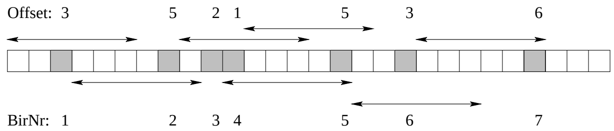


Figure 1: Bit selection when using a constant interval. A random number is computed within the interval to determine the next bit to be used.

- Extraction of redundant bits,*i.e.* those bits that can be modified without rendering the content of the cover medium useless.
- Selection of bits in which hidden information should be placed.

Separating the embedding process into two parts allows for easy replacement. A different data format can be accommodated with a different extractor algorithm, and new selection strategies can be implemented without changing other parts of the system. In addition, the computational complexity of the selection does not depend on the cost of the extraction.

One valid objection against this separation is the potential loss of information that might be helpful in the selection step. For example, an image may have areas of high complexity that can either hold more hidden information or in which modifications are less likely to be detected. In our model, the selection algorithm see only the redundant bits and is not aware of their origin. To remedy this, the extractor adds attributes to each redundant bit. These attributes indicate if a bit is locked or how detectable changes to it are.

## 4.2 Extraction of Redundant Bits

In general, extracting the redundant bits of a data source depends heavily on the specific data format. One has to be aware that the embedding actually happens when the cover medium is written out in its specific data format. Conversion to the final data format might include operations like compression, and is not necessarily deterministic. To reduce modifications to the cover medium, we require knowledge of the redundant bits before the actual stego medium is created. One way to achieve this is to perform all operations involved in creating the output object, and to save the redundant bits encountered. *E.g.* for the JPEG image format this might be the LSB of the discrete cosine transform coefficients.

The redundant bits and the information hidden in them are written back when the final output is created. This requires determinism in the conversion process, which can always be ensured by replacing random processes with a pseudo-random number generator that is initialized to the same state for the extraction and the final conversion step.

Before the extracted bits are passed to the selector, they are annotated with additional information. This information includes locked bits, *i.e.* bits that may not be modified in the embedding process, and a heuristic that determines how detectable changes to a bit might be.

A bit is locked when the bit has already been used to carry hidden information. This can occur when more than one message is hidden in the cover medium.

## 4.3 Selection of Bits

Before the selection of redundant bits can begin, an RC4 stream cipher [10] is initialized with a user-chosen secret key. From the stream cipher we derive a pseudo-random number generator (PRNG) for the selection process and also use it to encrypt the hidden message. The bits that are replaced with information from the hidden message are selected with the help of the pseudo-random number generator as follows.

First, we need to hide 32 state bits. The state is a concatenation of a 16-bit seed and a 16-bit integer containing the length of the hidden message. Selection starts at the beginning of the extracted bits. We determine the next bit by computing a random offset within a fixed interval and adding that offset to the current bit position. To compute the random offsets, we use the pseudo-random number generator described earlier. The data at the new bit position is replaced with message data. This process is iterated 32 times. The resulting bit positions can be represented as,

$$\begin{aligned} b_0 &= 0 \\ b_i &= b_{i-1} + R_i(x) \quad \text{for } i = 1, \ldots, n \end{aligned}$$

where $b_i$ is the position of the $i$-th selected bit, and $R_i(x)$ is a random offset in the interval $[1, x]$. This process is illustrated in figure 1.

After the state data has been embedded, the pseudo-random number generator is reseeded with the 16-bit seed. The remaining length of the hidden message is used to adapt the interval out of which the random numbers are drawn to the amount of remaining data,

$$\text{interval} \approx \frac{2 \times \text{remaining bits in bitmap}}{\text{remaining length of message}}.$$

The selection process continues as outlined above, the only difference being that the interval is adjusted every eight bits. This ensures that the hidden message is evenly distributed over all available bits.

The same selection process is used when retrieving a hidden message from a stego medium. The interval size is changed only after the state has been embedded, so the state is retrievable and can be used to reseed the pseudo-random number generator correctly.

### 4.4 Riding the Gaussian Distribution

We now explain how modifications to the cover medium can be reduced with the selection algorithm from the previous section.

In the embedding process, the 16-bit seed can be freely chosen. Each different seed creates a new pseudo-random number generator and thus a different selection of bits. In effect, we have created a family of independent pseudo-random number generators. When measuring the distribution of changed bits that each PRNG creates in the embedding process can be modeled by a Gaussian distribution,

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2},$$

where $\mu$ is the mean number of changed bits, and $\sigma$ is the standard deviation of the distribution.

As the hidden data has been encrypted by the RC4 stream cipher, it has the properties of a random stream of bytes, so we expect the mean $\mu$ to be close to 50% of the bits in the hidden data.

Picking the seed that represents the changed bits at the lower end of the Gaussian distribution, allows us to reduce the number of bits that have to be modified; Figure 2. It becomes harder to detect the modifications as more of the hidden message is already naturally represented in the redundant bits.

Detectability is also used as a bias in the selection process. The selector does not try only to minimize the number of changed bits but also the overall detectability. Whenever a bit has to be modified, its detectability will be added to a global bias. A higher accumulated bias reduces the likelihood that this specific embedding will be used.

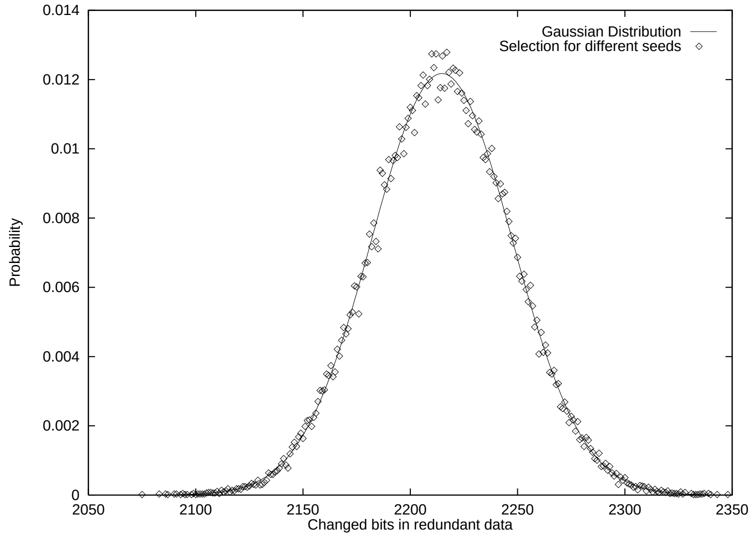


Figure 2: Probability distribution of changed bits for different seeds compared to a Gaussian distribution.

| Mean $\mu$ | Standard Deviation $\sigma$ | $\frac{\sigma^2}{\mu}$ |
|---|---|---|
| 2514.942 | 33.761 | 0.45321 |
| 3277.705 | 38.564 | 0.45374 |
| 5988.734 | 52.275 | 0.45631 |
| 7643.959 | 59.141 | 0.45758 |
| 9023.823 | 64.005 | 0.45398 |
| 11192.743 | 71.674 | 0.45898 |

Figure 3: The mean $\mu$ and standard deviation $\sigma$ for the embedding of different sized messages into a 106317-bit bitmap extracted from a JPEG-image.

For a given cover medium, we observe that the ratio of the squared standard deviation and the mean remain constant for different hidden messages,

$$\frac{\sigma^2}{\mu} \approx const,$$

as shown in Figure 3. As a result, more modifications can be avoided percentage wise by reseeding the iterator for smaller hidden messages than for bigger ones.

Furthermore, by varying the seed the algorithm is able to find an embedding that does not have conflicts with locked bits. This allows us to hide multiple data sets within a cover medium, which we discuss next.

## 4.5 Choices with Coding Theory

While the seeding allows us to view the embedding as a probabilistic process, the flexibility of the PRNG family in selecting bits is not always enough to avoid all bits that are locked or have a high detectability. We could prevent modification of those bits if it were possible to introduce errors into our hidden message without destroying its content. In other words, all introduced errors need to be correctable.

Coding theory provides us with codes that can correct errors by maximum-likelihood decoding. We write $[n, k, d]$ to indicate a $k$-dimensional linear code of length $n$ with Hamming distance $d$. Such a code can correct $t$ errors, where $d = 2t + 1$.

In general, the application of an error-correcting code increases the data that we need to embed. A $k$-bit data block will result in an $n$-bit code block. However, we first observe that about one half of the data is already represented and does not need to be modified, and second, that we can introduce $t$ additional errors in the code block. Thus, if

$$\frac{n}{2} - t = \frac{k}{2} \tag{1}$$

holds, we expect the number of modifications for the encoded data to be the same as for the unencoded data. Equation (1) can be rewritten as

$$d = n - k + 1,$$

which is exactly the Singleton bound fulfilled by all maximum distance separable (MDS) codes [11].

Unfortunately, the only non-trivial binary MDS code is the repetition code $[n, 1, n]$. It's major drawback is the $n$-fold repetition of the data, so that it is useful only for small hidden messages.

Once the data is encoded, we can choose $t$ bits in each code block that do not need to be modified in the redundant data. Our choice is determined by the conflicts present in the block, and after all conflicts have been resolved by the detectability of changes.

### 4.6 Plausible Deniability

To embed a hidden message into the cover medium, we modify the redundant data of the cover medium. The redundant data might have properties of a statistical nature of which we are not aware, or understand less than an adversary. If the embedding process changes the characteristics of the cover medium, a more knowledgeable observer can conclude the presence of a hidden message without necessarily being able to point to the specific bits that were changed.

The originator of the stego medium might now be forced to reveal the hidden communication. However, we assume that the only predicate the observer can ascertain is the fact that the cover medium was modified. If the sender embeds multiple hidden messages into the cover medium, he can include an innocuous message, turn that over on request, claim that there is no other information hidden in the stego medium, and leave unharmed. This is called *plausible deniability.*

Actually, the described mechanism already implicitly supports plausible deniability. More than one hidden message can be embedded, as the "locked bit" attribute prevents information from other hidden messages from being overwritten. Depending on the size of the hidden messages, the likelihood that a selection is found that does not conflict with previously locked bits can be small. In that case, error-correcting codes can be employed to increase the selection flexibility.

### 4.7 Hidden information dictates cover medium

Usually, a hidden message is embedded into a specific medium. Instead of selecting a specific cover medium, the hidden message can be examined and a cover medium will be selected that allows embedding with minimal modifications.

This can be achieved by embedding the hidden message into multiple cover media. Afterwards, the cover medium that results in the fewest modifications is chosen.

| Method | Correlation to One | Maurer Test |
|---|---|---|
| None | $63.77\% \pm 3.37\%$ | $6.704 \pm 0.253$ |
| Best | $59.10\% \pm 3.19\%$ | $6.976 \pm 0.168$ |
| Worst | $59.06\% \pm 3.16\%$ | $6.978 \pm 0.169$ |

Figure 4: Comparison between different selections and the resulting embeddings.

As expected, the distribution of changed bits in the different cover media follows a Gaussian distribution similar to the one shown in section 4.4.

# 5 Analysis for JPEG

Although the methods in this paper do not depend on the actual data format of the cover medium, we restrict our analysis to JPEG images, a very common data format.

For testing, we embed data into 54 pictures that were taken with a Fuji MX-1700 digital camera around Ann Arbor, Michigan. The size of the images is $640 \times 480$ pixels. After the images were downloaded from the camera, they were recompressed with a quality factor of 75. This simulates the conversion step in the embedding process without actually embedding any data.

The least-significant bits of the discrete cosine transform coefficients were passed as redundant bits to the selector process. The average number of redundant bits for this set of images is about $46,000$, varying between $30,000$ and $97,000$.

Without modifying the redundant bits, we notice a strong correlation towards one. About 63.8% of all the bits are set with only a small standard deviation of $\pm 3.4\%$ between images.

The statistical attack described by Westfeld and Pfitzmann applies only to embedding methods that do not spread the hidden message over all the redundant bits [14]. As our methods are different in that regard, we need to find another way to evaluate the redundant bits.

For that, we test the data with Ueli Maurer's "Universal Statistical Test for Random Bit Generators" [6] using a block size of eight bits. The measured entropy for the extracted redundant bits is $6.704 \pm 0.253$. The expected value for a truly random source is 7.184.

To compare the differences in the selection process, we embed the first chapter of Lewis Carroll's "The Hunting of the Snark" into the images. After compression, the hidden message has a size of about $14,700$ bits. The selection process is configured to find the best and the worst embedding. Examining the results from all 54 images, the best embedding modifies only 7089 bits and the worst 7649 bits. That is a difference of 560 bits, or about 3.8% of the hidden message size.

Figure 4 shows the results from our analysis. We note that for the best selection as well as for the worst selection the correlation to one for the redundant bits is lower than for the unmodified images. Compared to the best case, the worst selection has a slightly lower correlation to one, but the difference is not significant.

On the other hand, the Maurer test shows as expected a higher randomness in the redundant bits after the hidden message has been embedded. In this case, the worst selection is slightly more random than the best.

While the selection process does not seem to make a large difference for the presented statistical tests, the probabilistic embedding might still be helpful in conjunction with other methods.

## 6 Conclusion

The steganographic embedding process can be separated into two distinct steps: Extraction of redundant bits and selection of bit positions that are used to carry the hidden message.

We introduce two new methods to improve the selection process. The first uses a seeded pseudo-random number generator to determine the fewest modifications to the cover medium. The second uses error-correcting codes to increase the flexibility in selecting bits without increasing the number of necessary changes.

Together, these methods can be used to provide plausible deniability be embedding multiple hidden messages in the cover medium.

The selection and encoding methods have been implemented in the OutGuess [9] program which is freely available as source code.

## 7 Acknowledgments

I thank Andrew Carra for earlier discussions on minimizing modifications to the cover medium, and Sean Coffey for helpful comments on error-correcting codes. I also thank my advisor Peter Honeyman for careful reviews and suggestions.

## References

[1] Thomas Aura. Practical Invisibility in Digital Communication. In *Proceedings of Information Hiding - First International Workshop*. Springer-Verlag, May/June 1996.

[2] J. Zöllner et al. Modelling the Security of Steganographic Systems. In *Proceedings of Information Hiding - Second International Workshop*. Springer-Verlag, April 1998.

[3] Oded Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudo-randomness*. Springer-Verlag, 1999.

[4] Neil F. Johnson and Sushil Jajodia. Steganalysis of Images Created Using Current Steganographic Software. In *Proceedings of Information Hiding - Second International Workshop*. Springer-Verlag, April 1998.

[5] Lisa M. Marvel, Charles G. Boncelet, Jr., and Charles T. Retter. Reliable Blind Information Hiding for Images. In *Proceedings of Information Hiding - Second International Workshop*. Springer-Verlag, April 1998.

[6] Ueli M. Maurer. A Universal Statistical Test for Random Bit Generators. *Journal of Cryptology*, 5(2):89–105, 1992.

[7] W. W. Peterson and E. J. Weldon. *Error-Correcting Codes*. MIT Press, second edition, 1972.

[8] Birgit Pfitzmann et al. Information Hiding Terminology. In *Proceedings of Information Hiding - First International Workshop*. Springer-Verlag, May/June 1996.

[9] Niels Provos. OutGuess - Universal Steganography. `http://www.outguess.org/`, August 1998.

[10] RSA Data Security. The RC4 Encryption Algorithm, March 1992.

[11] J. H. van Lint. *Introduction to Coding Theory*. Springer-Verlag, 2nd edition, 1992.

[12] G. W. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, April 1991.

[13] Steve Walton. Information Authentication for a Slippery New Age. *Dr. Dobbs Journal*, 20(4):18–26, April 1995.

[14] Andreas Westfeld and Andreas Pfitzmann. Attacks on Steganographic Systems. In *Proceedings of Information Hiding - Third International Workshop*. Springer Verlag, September 1999.