CHAPTER 12

Steganography

Steganography is the act of covert communications, which means that only the sender, Alice, and receiver, Bob, are aware of the secret communication. To accomplish this, the secret message is hidden within benign-looking communications known as covertexts or cover Works. To an adversary, Eve, it is clear that Alice and Bob are communicating, but the combined covertext and hidden message, referred to as a stegotext or stego Work, appears to be innocuous (i.e., Eve is unaware that the innocuous content hides a message).

The main requirement of steganography is *undetectability*, which, loosely defined, means that no algorithm exists that can determine whether a Work contains a hidden message. Steganalysis is the process of detection of steganographic communications. And since steganography and steganalysis are closely intertwined, some of the following discussion will oscillate between one and the other.

Steganography and watermarking are both forms of data hiding and share some common foundations. Nevertheless, it is worth reiterating the goals of these two data-hiding applications in order to highlight the key differences.

In Chapter 1, we defined watermarking as the practice of imperceptibly altering a Work to embed a message about that Work. This message might, for example, be a digest or hash of the Work, ownership information, a unique identifier of the sender or the consumer, or digital rights management information. However, whatever the message, the number of bits needed to be embedded is relatively small—typically 8 bits and almost never more than 128. The fact that a Work contains a watermark is often widely known. Indeed, in many applications, knowledge of the use of watermarking is actively promoted to deter and discourage illegal use of the cover Work. And it is the cover Work that is of principal value, thereby requiring that the watermark be imperceptible.

> *We define steganography as the practice of undetectably altering a Work to embed a message.*

Comparing this definition with that of watermarking, we note that we no longer require imperceptibility, but rather undetectability. Thus, in theory, the change

may be perceptible, as for example, when an image of a man in a blue suit is altered to one of a man in a black suit. Perceptibility is permitted because the cover Work has no intrinsic value. And a perceptible change may still be undetectable (i.e., no algorithm exists that is able to determine whether a Work contains a hidden message), since the adversary does not have access to the original, unmodified cover Work. Since the message no longer needs to be "about that Work," but rather is arbitrary, the desired length of the message may be very much longer than for watermarking applications—it is not uncommon for steganographic algorithms to embed thousands of bits. Fortunately, this requirement is made somewhat easier by the fact that Alice is free to select which cover Work to use. Thus, Works in which it is difficult to conceal a message can be avoided. However, digital watermarking has no such control over the selection of Works.

When designing a steganographic scheme, we need to consider issues such as the properties of the communication channel, the source of cover Works, and the embedding/extraction function. In Sections 12.1 and 12.2, we discuss these issues and define basic terminology that is commonly used to describe and evaluate steganographic schemes.

The central concept in steganography is statistical undetectability. Without a precise definition, the field of steganography would lack the criterion to evaluate how secure steganographic schemes really are. In Section 12.3, we provide an information-theoretic framework for steganography, which provides a rigorous definition for *undetectability*. The information-theoretic concepts are illustrated using one of the simplest algorithms for digital steganography, which hides a message in the least significant bit (LSB) of a Work.[1]

The theoretical foundations described in Section 12.3 are reflected in Section 12.4, where we discuss practical steganographic schemes that aspire to satisfy the requirements for statistical undetectability. We describe a general framework for constructing steganographic schemes based on a statistical model of the cover Works. We also mention other approaches that attempt to build statistically undetectable schemes using heuristic principles, such as masking the embedding modification as a naturally ocurring process.

In Section 12.5, we cast the embedding process in terms of coding theory. This formulation enables us to minimize the number of embedding changes (matrix embedding) and construct steganographic schemes with arbitrary selection rules (wet paper codes) (i.e., the detector does not need to know where the embedder altered the cover Work).

The complementary task to steganography is steganalysis—discovering the *presence* of steganographic channels. Steganography is considered broken if even the presence of secretly embedded data is detected (i.e., it is not

[1] As with almost all of this book, we focus on digital images, but the concepts can be easily generalized to other forms of digital content.

necessary to decode the message). The task of recovering some attributes of the message or the stego method used is called *forensic steganalysis*. Steganalysis can be divided into two categories—targeted and blind—depending on whether the attack uses the knowledge of the steganographic algorithm. The subject of steganalysis is treated in Chapter 13.

12.1 STEGANOGRAPHIC COMMUNICATION

The first informal definition of a steganographic scheme was formulated by Simmons [375] as the Prisoners' Problem. Two prisoners, Alice and Bob, are under the surveillance of a warden, Eve. The warden will permit Alice and Bob to communicate, but all communications must go through the warden. If the warden thinks that Alice's message to Bob is innocuous, she may simply forward it to Bob. Alternatively, she may intentionally distort the content (e.g., apply lossy compression) in the hope that such a distortion will remove any secret message that just might be present. If the warden thinks Alice's message to Bob hides a covert communication, then she may block the communication entirely.

This framework, which models the applications discussed in Chapter 2, is depicted in Figure 12.1. A number of different assumptions can be made regarding the channel, the source of cover Works, and the embedding and extraction functions. In the next section, we elaborate on the properties of the channel, while the subsequent section focuses on the basic properties of the embedding and extraction functions.

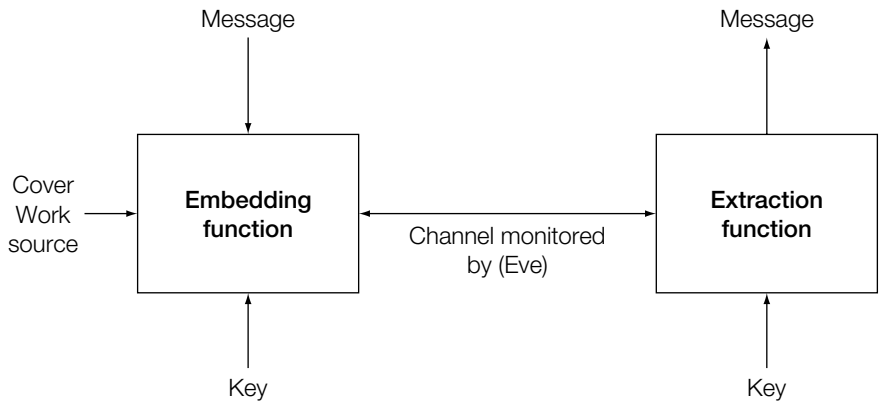

FIGURE 12.1

Steganographic embedding scheme.

12.1.1 The Channel

In steganography, the physical channel used for communication is generally assumed noise free, as this can be ensured using error correction and standard Internet protocols. Instead, the channel's properties are defined by the warden. The warden is considered a part of the channel because she may, or may not, interfere with the communication. As such, there are three types of warden: passive, active, and malicious.

The warden is called *passive* if she is restricted from modifying the content sent by Alice prior to receipt by Bob (i.e., the warden can only prevent or permit delivery of Alice's message). In this scenario, the warden tests each communication from Alice for the presence of a covert message. If the warden's test is negative, the communication is relayed to Bob. Otherwise it is blocked. This is the most commonly assumed scenario and why most steganographic algorithms are not designed to be robust.

The warden is called *active* if she intentionally modifies the content sent by Alice prior to receipt by Bob. In this scenario, the warden may not be entirely confident of her steganalysis program. Thus, even though her tests are negative, the warden may alter the content, hoping that the modification will destroy any steganographic message that *might* be present. If the steganographic algorithm assumes a passive warden, then there is a good chance that alterations to the content will severely degrade or remove the hidden message. The types of modification an active warden might apply include lossy recompression of images and audio clips, low-pass filtering, and other procedures that slightly degrade the content.

The warden is called *malicious* if her actions are based on the specifics of the steganographic scheme and are aimed at catching the prisoners communicating secretly. This may include the warden trying to impersonate Alice or Bob or otherwise tricking them. A malicious warden is usually considered in public-key steganography [22, 430]. In this scenario, the *stego* key is known and anyone can extract the secret message. However, the message is encrypted using a public-key cryptosystem. Only those who possess Bob's private key can decipher Alice's message. Even though the stego key is known, it is difficult to distinguish between an encrypted message and a random bit sequence extracted from a cover Work. Nevertheless, since the Warden also knows the stego key, she has more options to attack the stego system [26, 100].

In this chapter we will not consider cases where the warden is active or malicious, since the vast majority of research in steganography is concerned with the passive warden scenario. We focus on steganography in digital media, because this field is the most developed today, ignoring other less-developed topics, such as linguistic steganography [441] or data hiding in network protocols [277]. The subject of subliminal channels is covered in the cryptographic literature [374].

In the next section we describe various choices Alice and Bob have when designing a steganographic scheme. We also define basic terminology that is commonly used to describe and evaluate steganographic schemes.

12.1.2 The Building Blocks

The main building blocks of any steganographic algorithm are:

1. The choice of the cover Work.
2. The embedding and extracting algorithms, which might include
   a. Symbol assignment function.
   b. The embedding modification.
   c. The selection rule.
3. Stego key management.

We now discuss each of these design elements in more detail.

Unlike a watermark, a steganographic message says nothing about the cover Work in which it is hidden. Consequently, the steganographer is free to choose a particular cover Work from his or her source of covers. The main restriction is the source of cover Works, which is determined by the resources available to Alice and Bob, by the warden herself, and the context in which the communication takes place. For example, an oppressive regime (Section 2.2.1) can specify allowable forms of messages and Alice must comply with them to avoid being caught. Or, if Alice and Bob communicate by posting images to a discussion newsgroup, they must choose the covers among those that are typically posted. But even with these restrictions, there are still numerous cover Works in which to hide the covert message. Alice is therefore at liberty to choose the cover Work which, after embedding, has the least likelihood of being detected. For example, it is intuitively clear that noisy or highly textured images will better mask any embedding changes than high-quality images with little content (e.g., blue sky images). Alternatively, Alice can think ahead and attempt to guess what tests the warden is going to use and embed the same message into many different covers, run known steganalysis attacks on each stego Work, and then simply send the cover that passes the tests [233].[2]

[2] In fact, it is possible to go a step further and choose the cover Work such that it is correlated with the hidden message. For example, if the hidden message is an image, choose an image that is similar. The advantage of this is that the minimum number of bits needed to encode the hidden message is now the conditional entropy of the hidden image given the cover. This may be very much smaller than the entropy of the hidden image itself. As we will see, the fewer bits that we need to hide, the less likely the warden will detect the stego Work. Furthermore, the information transferred by Alice to Bob can now be much greater than the number of bits embedded. This is because the cover Work is now providing Bob with additional information. The interested reader is directed to [92, 460] for further details.

Given the source of cover Works (e.g., a class of images or songs), Alice and Bob need to construct the embedding and extraction functions. In watermarking, it is not uncommon to specify the detector design but not the encoder design, since there may be a variety of ways to perform the encoding, each with different tradeoffs (e.g., computational and economic costs). This design principle is absent from steganography, where both the embedding and extracting functions are jointly defined.

Fundamentally, an embedding function can be based on three different principles, namely:

1. The cover Works are preexisting and the embedder does *not* modify the cover Works. This is referred to as steganography by cover lookup.
2. The cover Works are generated based on the hidden message and the embedder does *not* modify the cover Works. This is referred to as cover synthesis.
3. The cover Works are preexisting and the embedder modifies the cover Works. This is referred to as steganography by cover modification.

How can we send a message without modifying the cover Work? Consider the case where Alice only needs to send, say, 10 bits to Bob. Then, if Alice has a collection of approximately 1,000 songs, she can determine which song, concatentated with their shared key, hashes to the desired 10-bit message. Hence, steganography by cover lookup. Since the song has not been modified in any way, it is almost impossible for the warden to determine that a steganographic message is present. Of course, as the required payload increases, this solution quickly becomes impractical. To send 20 bits requires a million songs, and to send 30 bits requires a billion songs. Although Alice might solve this problem by sending 10k bits at a time, a 10k-bit message would require the transmission of 1,000 songs, which might raise the suspicion of the warden.

In steganography by cover synthesis, Alice *creates* the stego Work without recourse to a cover Work. An interesting real-world example of such a system has been described at the end of *Between Silk and Cyanide*, by Leo Marks [284]. The code, called "Windswept," which was used by British spies in World War II, works in the following manner. The code consisted of a big book of conversations with alternate wordings for each line, and even alternate courses that the conversation could take. Each line was associated, arbitrarily, with a number. By selecting different phrases from the book, British spies could thus encode sequences of numbers in perfectly innocuous conversations. The code had fairly high capacity because the communicating parties did not care at all about the content of the conversations and thus could change it in any way necessary to suit the secrets.

Another method to synthesize the stego Work uses so-called mimic functions [440]. The program SpamMimic (*http://www.spammimic.com*) will encode a short message into a document that resembles a typical spam. Basically, the sentences produced by the program are a function of the secret message. Because spammers use unusual spellings, punctuation, and sentence structures, it is rather hard to distinguish spam containing a hidden message from regular spam.

A further example of steganography by synthesis is called data masking [341]. Here the goal is to shape a secret message into a signal whose statistical properties resemble those of, say, music. If the signal indeed shares essential statistical properties with typical music files, then any automatic steganalysis engine that checks for statistical anomalies will not detect anything suspicious. Of course, this method of steganography will not be successful if the warden listens to the music that Alice and Bob exchange.

Steganography by cover modification describes methods where Alice alters an existing cover Work to create a stego Work that conveys the desired message. This approach is both the most common and the most advanced. In this book, we will only focus on this class of steganographic algorithms.

The type of changes introduced by the embedder, together with the location of these changes within the cover Work, have a major influence on how inconspicuous the embedded message will be. Intuitively, changes of large magnitude will be more obvious than changes of smaller magnitude. Consequently, most steganographic schemes try to modify the cover Work as little as possible.

The location of the changes is controlled by the *selection rule*. There are three types of selection rules: sequential, (pseudo) random, and adaptive.

A sequential selection rule embeds the message bits in individual elements of the cover Work in a sequential manner, for example, starting in the upper left corner of an image and proceeding in a row-wise manner to the lower right corner. Although the sequential selection rule is the easiest one to implement, it provides poor security, since steganalysis algorithms can inspect the statistical properties of pixels in the same order, looking for a sudden change in behavior (see Section 13.2.1).

A pseudo-random selection rule embeds the message bits in a pseudo-randomly selected subset of the cover Work. The sender might first use a secret stego key, K_s, to initialize a pseudo-random number generator (PRNG) that in turn generates a pseudo-random walk through the cover Work. The message bits are then embedded into the elements constituting this walk. Pseudo-random selection rules typically offer better security than sequential rules.

An adaptive selection rule embeds the message bits at locations that are determined based on the content of the cover Work. The motivation for this is that statistical detectability is likely to depend on the content of the cover

Work as well. For example, it will be more difficult to detect embedding changes in noisy images or in highly textured areas of the image compared with smooth, uniform areas. Thus, one may desire to adjust the selection rule to the specific content of the cover Work. For example, consider LSB embedding once more. The selection rule could depend on the variance of pixels within a small local neighborhood. Only pixels whose local neighborhood variance exceeds a certain threshold would be candidates to be modified. The influence of the selection rule on the security of steganographic schemes are discussed in more detail in Section 12.5 where we introduce wet paper codes.

The process of embedding is controlled by a secret key shared between Alice and Bob. The key can be used for several different purposes. As previously mentioned, the key may seed a pseudo-random number generator to generate a random walk through the cover Work. It can also be used to generate other pseudo-random entities needed for embedding. For example, in stochastic modulation steganography (see Section 12.4.3), the message is embedded by adding a noise signal with specific statistical properties to the cover Work. The generation of this signal may depend on the stego key.

Alice and Bob do not have to use, and in fact should not use, the same stego key for every cover. Imagine that the covers are all images of the same dimensions. Then, a fixed stego key would produce the same pseudo-random embedding walk in every cover. The warden could use the fact that the embedding changes between different covers are correlated to mount an attack. Thus, Alice and Bob may wish to adopt a more sophisticated key management and periodically change the key according to some pre-agreed protocol. For example, the message may be communicated using a session key that is different for each cover and communicated in the cover itself.

It is important to choose strong stego keys, otherwise the warden could attack the steganographic scheme simply by trying to read messages from the stego Work using all possible stego keys. The correct key would be revealed when a meaningful message is obtained. Although this attack would not work if the message was encrypted before embedding, there exist more advanced versions of this attack [151]. In general, it is always a good practice to encrypt the message before embedding. The crypto key could be derived from the stego key or could be chosen independently. The second choice provides better security in cases where the stego key is compromised.

The primary goal of steganography is to design embedding functions that are statistically undetectable and capable of communicating practical (i.e., large) payloads. In order to do so, we need to first define in mathematical terms what we mean by statistical undetectability. This is the subject of Section 12.3. However, before proceeding to do so, we briefly introduce some notation and terminology.

12.2 NOTATION AND TERMINOLOGY

We now mathematically define a steganographic scheme. Let K_s denote a stego key drawn from a set, $\mathcal{K}$, of all secret stego keys, $\mathcal{M}$ the set of all embeddable messages, and $\mathcal{C}$ the set of all cover Works. A steganographic scheme is formed by two mappings, the embedding mapping, *Emb*, and the extraction mapping, *Ext*:

$$Emb: \mathcal{C} \times \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$$

$$Ext: \mathcal{C} \rightarrow \mathcal{M}, \tag{12.1}$$

such that $Ext(Emb(\mathbf{c}, K_s, \mathbf{m})) = \mathbf{m}$ for all $\mathbf{c} \in \mathcal{C}$, $K_s \in \mathcal{K}$, and $\mathbf{m} \in \mathcal{M}$. The Work $\mathbf{s} = Emb(\mathbf{c}, K_s, \mathbf{m})$ is called the stego Work.

The embedding algorithm *Emb* takes the cover Work, the secret key, and the message as its input and produces the modified stego Work. For a given embedding function, the cardinality, $|\mathcal{M}|$, of the set, $\mathcal{M}$, is the number of different messages that can be embedded in a specific Work. The logarithm, $\log_2 |\mathcal{M}|$, is called the *embedding capacity* and its units are bits. For example, for LSB embedding in a grayscale image, the embedding capacity is equal to the number of pixels. If the length, m, of the embedded message is smaller than the embedding capacity, we say that the message has *relative length*, $m/|\mathcal{M}|$.

The impact of embedding (or the embedding distortion) is measured as $D(\mathbf{c}, \mathbf{s})$, where D is a distance defined on $\mathcal{C}$. A commonly used measure of distance is mean square error, as discussed in Chapter 8. Intuitively, steganographic schemes that introduce smaller embedding distortion are less likely to introduce statistically detectable artifacts than schemes with larger distortion. The average number of embedded bits per unit distortion is called the *embedding efficiency*. We study this concept in more detail in Section 12.5.

12.3 INFORMATION-THEORETIC FOUNDATIONS OF STEGANOGRAPHY

In 1949, Shannon [365] wrote a seminar paper entitled "Communications Theory of Secrecy Systems," within which he provided a precise definition of cryptographic security. Interestingly, Shannon identified three forms of secret communications: (1) "*concealment systems, including such methods as invisible ink, concealing a message in an innocent text, or in a fake covering cryptogram, or other methods in which the existence of the message is concealed from the enemy*," (2) privacy sytems, and (3) cryptographic systems. However, with regard to concealment systems (i.e., steganography), Shannon declared that such "*systems are primarily a psychological problem*" and did not consider them further.

Is it possible to give steganographic security the same precise meaning as Shannon gave to information security? In this section, we attempt to give the reader at least a partial answer to this question and to explain why the answer continues to be debated by researchers.

To the best of our knowledge, it was not until 1996 that the application of information theory to steganography was revisited. Anderson and Petitcolas [20, 22] observed that if perfect compression existed, then it would be possible to transform any arbitrary bitstream into a legitimate cover Work in the following manner. Consider a perfect compressor of music. If such a compressor existed, then every perceptually different music file would be represented by a unique code word. And these code words would appear to be random bit sequences. More important, the associated decompressor would convert any random bit sequence to a perceptually acceptable musical piece. Thus, if Alice passes her encrypted hidden message through this perfect *decompressor*, the result will be music. If the warden permits Alice to send any form of music to Bob, then no modifications to an existing piece of music are necessary. This is a form of embedding based on cover synthesis. Finally, Bob can compress the received music file through the perfect compressor in order to obtain the original encrypted message.

Of course, the chances of a random bit string mapping to a known musical piece is unlikely, and the warden may become suspicious if she does not recognize any of the musical pieces that Alice is sending to Bob. Notice that the fact that Alice and Bob never modify their Works does not necessarily imply that the stego method is undetectable. To explain why this does not have to be so, imagine that Alice sends only one out of a small number of possible messages. For example, either "attack" or "do not attack." Then, over time she would be resending the same pair of music files, which would be very suspicious. She can attempt to alleviate this problem by appending a string of dummy random bits to her messages so that the messages are mapped onto a larger set of musical files. However, even if Alice appends 128 random bits she still cannot be sure if the statistical distribution of her stego musical files is free of obvious artifacts. In other words, Alice needs to be concerned with the statistical distribution of her stego Works and how closely it matches some "natural" distribution. The question then becomes, how much information can Alice hide in a Work chosen from a distribution of cover Works, while ensuring that the probability of detection is negligibly small? This observation forms the foundation of the most widely used definition of steganographic security due to Cachin.

12.3.1 Cachin's Definition of Steganographic Security

Cachin's definition of steganographic security [60] assumes that the warden will permit Alice to send any cover Work, $\mathbf{c}$, to Bob, provided it is drawn from a probability distribution, P_C. And $P_C(\mathbf{c})$ is the probability of drawing a cover Work, $\mathbf{c}$, from this distribution.

Before proceeding, it is worth considering what this assumption means. Qualitatively, the probability distribution, $P_{\mathcal{C}}$, represents Eve's knowledge of what types of transmissions between Alice and Bob are legitimate. For example, it might be that Eve expects Alice and Bob to only transmit family photos. Thus, provided a photograph, $\mathbf{c}$, contains an image of one or more family members, then the image has a finite probability, $P_{\mathcal{C}}(\mathbf{c})$, of being generated from this distribution. Conversely, a photograph of Queen Elizabeth would have a very low or zero probability of deriving from the assumed distribution, $P_{\mathcal{C}}$, and therefore a very high likelihood, in Eve's mind, of being suspicious.

In practice, it may be very difficult to quantify this distribution of "natural" images. However, it may be feasible for some specific steganographic methods to describe the opposite—the distribution of Works carrying a secret message. In other words, even though we may not be able to tell if an image is "natural," we may be very certain that it is not, because it contains anomalies that are characteristic of a certain steganographic method. We illustrate this point with an example of the most commonly used steganographic method, called LSB embedding.

In LSB embedding, the message bits are encoded as the LSBs of pixel values.[3] For example, to embed a 0 at a pixel with grayscale value 51, we first write the value as a binary number $(51)_2 = 00110011$, the last bit being the LSB and the first bit the most significant bit (MSB), and then replace the LSB with the message bit. In this case, the binary representation is modified to $00110010 = (50)_2$. Obviously, if we were to embed a 1, no change would be necessary. As a result of our embedding, the value 51 was changed to 50. It is easy to see that the values 50 and 51 form a pair, such that during embedding these two values can be changed into each other but never to any other values. Thus, the set of all possible pixel values can be divided into disjoint pairs of values (PoVs), $(2i, 2i + 1)$, $i = 0, \ldots, 127$, that may be changed into each other during embedding.

The asymmetry of the embedding function introduces a characteristic artifact into the first-order statistics (the histogram) of the stego Work that practically never occurs in real imagery. In Figure 12.2, we show a portion of the histogram of a grayscale cover image before and after the cover has been fully embedded with LSB embedding (1 bit per each pixel). We can clearly see that the histogram bins of each PoV, $(2i, 2i + 1)$, have been equalized by embedding. This is, in fact, intuitively clear, because in an image fully embedded with LSB embedding, the LSBs have been completely randomized and thus any pixel from the PoV is equally likely to end up with value $2i$ as

[3] We note that LSB embedding can be applied to any collection of numerical samples, such as pixel values, audio samples, palette indices (for GIF images), or quantized DCT coefficients for JPEG images.

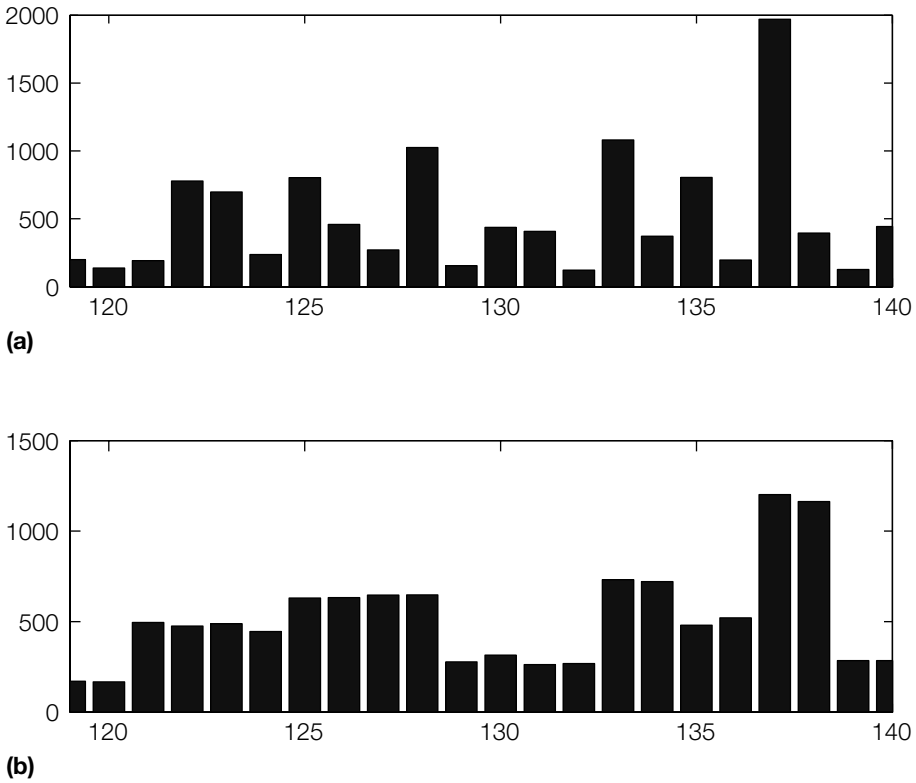

FIGURE 12.2

(a) A portion of a cover image histogram. (b) The same portion of the histogram of an image fully embedded with LSB embedding. Note the characteristic steps in the histogram.

with value $2i + 1$. This observation is the basis of the histogram attack described in Section 13.2.1.

Let us now try to mathematically formalize the concept of steganographic security. We will use the definition of a steganographic scheme given in Equation 12.1.

A steganographic scheme is formed by two mappings: the embedding mapping, *Emb*, and the extraction mapping, *Ext*. The embedding function assigns a stego Work, $\mathbf{s}$, to each combination of the cover Work, $\mathbf{c} \in \mathcal{C}$, stego key, $K_s \in \mathcal{K}$, and message, $\mathbf{m} \in \mathcal{M}$. Assuming the cover Work is drawn from the space of all possible cover Works with probability distribution, $P_{\mathcal{C}}$, and the stego key as well as the message are uniformly distributed over their corresponding spaces, $\mathcal{K}$, $\mathcal{M}$, the distribution of stego Works will be denoted as $P_{\mathcal{S}}$.

Cachin analyzed the warden's performance using the theory of hypothesis testing. Thus, given a Work, $\mathbf{c}$, Eve must decide between two hypotheses: H_0 represents the hypothesis that the Work *does not* contain a hidden message and H_1 represents the hypothesis that the Work *does* contain a hidden message. If hypothesis H_0 is true then the observation, $\mathbf{c}$, is drawn from the distribution, P_C. Conversely, if hypothesis H_1 is true then the observation, $\mathbf{c}$, is drawn from the distribution, P_S.

Comparison of the two distributions can be made based on their relative entropy, also known as the Kullback-Leibler distance, between two probability distributions, defined as

$$D(P_C \| P_S) = \sum_{\mathbf{c} \in C} P_C(\mathbf{c}) \log \frac{P_C(\mathbf{c})}{P_S(\mathbf{c})}. \tag{12.2}$$

The relative entropy is always non-negative and is equal to 0 if and only if $P_C = P_S$. Also note that for probability distributions on a set containing only two elements, $P = (p_1, 1 - p_1)$ and $Q = (q_1, 1 - q_1)$, Equation 12.2 becomes the binary relative entropy defined as

$$D(P \| Q) = p_1 \log \frac{p_1}{q_1} + (1 - p_1) \log \frac{1 - p_1}{1 - q_1}. \tag{12.3}$$

Although relative entropy is not a distance in the strict mathematical sense, because it is nonsymmetrical and does not satisfy the triangle inequality, it is useful to think of it as a distance. The relative entropy can be thought of as the difference between Huffman coding a source drawn from a pdf, P, using a table determined by the true distribution, P, and an alternative table determined by another distribution, Q (the latter leads to a suboptimal choice of code word lengths).

When $D(P_C \| P_S) = 0$, that is, the distribution of the stego Works, P_S, created by Alice is identical to the cover distribution, P_C, assumed by the warden, Alice's stegosystem is perfectly secure, as defined by Cachin. This is because it is impossible for the warden to distinguish between cover Works and stego Works. If $D(P_C \| P_S) \leq \epsilon$, then Cachin defined the system as ϵ-secure.

To understand what is meant by ϵ-secure, consider that Eve, the warden, can make two forms of error. The first type of error (type I) is a false positive that occurs when Eve decides that a hidden message is present when in fact it is absent. The second type of error (type II) is a false negative that occurs when Eve decides that a hidden message is absent, when in fact it is present. Let α and β denote the probabilities of type I and type II errors, respectively.

The response of Eve's detector is binary—it answers either 0 for cover or 1 for stego. Thus, assuming the detector receives only cover Works with probability P_C, Eve will decide 0 or 1 with probabilities $p_1 = 1 - \alpha$ and $p_2 = \alpha$, respectively. On stego Works distributed according to P_S, Eve's detector assigns 0 and 1 with probabilities $q_1 = \beta$ and $q_2 = 1 - \beta$, respectively.

The relative entropy given by Equation 12.3 between both distributions of Eve's detector is

$$d(\alpha, \beta) = (1 - \alpha) \log \frac{1 - \alpha}{\beta} + \alpha \log \frac{\alpha}{1 - \beta}. \tag{12.4}$$

A standard result in information theory states that the relative entropy of processed data can never increase (see, for example, Chapter 2 of Cover and Thomas [89]). Because Eve's detector is a type of processing, we must have

$$d(\alpha, \beta) \leq D(P_C \parallel P_S). \tag{12.5}$$

This inequality can be used to determine a lower bound on the probability of false negatives (type II error), β, given a desired upper bound on the probability of a false positive (type I error), α. In particular, if the probability of a type I error is $\alpha = 0$ (i.e., Eve is not permitted to accuse Alice of transmitting a covert message when in fact she has not), then setting $\alpha = 0$ in Equation 12.5 gives $\log \frac{1}{\beta} \leq D(P_C \parallel P_S) \leq \epsilon$. In other words, the probability of a type II error, β (i.e., of missing a covert communication) is

$$\beta \geq 2^{-\epsilon}. \tag{12.6}$$

Thus, the smaller ϵ is, or the closer the two distributions, P_C and P_S, are, the greater the likelihood that a covert communication will *not* be detected.

Steganographic systems that are secure in Cachin's sense indeed exist. Consider the following simple one-time pad steganographic system with $\mathcal{C} = \{0, 1\}^n$ and P_C the uniform distribution on $\mathcal{C}$. Given a secret message, $\mathbf{m} \in \{0, 1\}^n$, the sender selects $\mathbf{K}_s \in \mathcal{K} = \{0, 1\}^n$ at random and computes the stego object as the XOR (exclusive-or) $\mathbf{s} = \mathbf{K}_s \oplus \mathbf{m}$. The message is extracted by the recipient as $\mathbf{m} = \mathbf{K}_s \oplus \mathbf{s}$. This system is, however, not very useful because no warden will allow the exchange of random bit strings. For construction of other provably secure steganographic schemes, the reader is referred to [201, 234].

The information-theoretic definition of steganographic security does not consider issues of practical realizability and computational complexity. One could base the concept of steganographic security on the inability of the warden to carry out the necessary calculations to gather sufficient evidence that Alice and Bob communicate secretly. The definition proposed by Katzenbeisser and Petitcolas [222] takes into account computational complexity and is based on a probabilistic game played between a third party (a judge) and the warden. Another possibility is to define steganographic security with respect to a specific detector [70]. The capacity of steganographic channels with noise was studied in Harmsen and Pearlman [175]. Assuming there exist bounds on the maximal admissible distortion introduced by both the warden, Alice, and Bob, the problem of how much information can be reliably and securely communicated can be formulated in terms of game theory [131, 300, 303, 304, 436].

It is likely that the concept of steganographic security will follow the same path as security in cryptography. The only unconditionally secure cryptographic system is the one-time pad, which is, however, rarely used because of its impracticality. Cryptographic schemes, such as DES (Data Encryption Standard), cannot be proved secure but are nevertheless widely used because of their practical advantages. Their security lies in the fact that nobody has so far been able to produce an attack substantially faster than brute-force search for the key. Similarly, we may consider a steganographic system "practically secure" if no existing attacks can be modified to mount a successful attack on the steganographic scheme. In practice, security of specific embedding algorithms is usually evaluated using existing targeted staganalyzers and blind steganalyzers (see Section 13.1.1).

12.4 PRACTICAL STEGANOGRAPHIC METHODS

All practical steganographic methods try, in one way or another, to comply with Cachin's definition of steganographic security. This can be achieved either by postulating some heuristic principles and designing an embedding scheme that follows these heuristics or by finding a simplified model for the space of Works, $\mathcal{C}$, and making sure that the steganographic embedding preserves this model. We start with the second approach.

12.4.1 Statistics Preserving Steganography

Because the dimensionality of the space of Works, $\mathcal{C}$, is approximately determined by the number of pixels, it is very difficult to obtain even a rough approximation to the distribution, $P_{\mathcal{C}}$. We can, however, attempt to represent each Work in a much lower dimensional space. For example, we can assume that each Work is completely described by its histogram. This would, indeed, be the case if the colors in an image were realizations of independent indentically distributed (i.i.d.) random variables, which, of course, they are not. Under this assumption, a steganographic system that preserves the histogram of the cover Work would be secure.

In the next section, we describe one of the first steganographic schemes that aims to exactly preserve the histogram of 8×8 DCT coefficients in a cover Work. Because DCT coefficients in an individual 8×8 block are largely decorrelated and because interblock dependencies among DCT coefficients are less strong than dependencies among neighboring pixels in the spatial domain, the DCT coefficients can be approximately modeled as an i.i.d. signal. Of course, if the warden has a more sophisticated model, then preserving the first-order statistics of DCT coefficients may not be sufficient to avoid detection, as shown in Section 13.2.3.

Preserving DCT Statistics: OutGuess

The OutGuess algorithm [338] is a two-pass procedure. In the first pass, OutGuess embeds message bits along a pseudo-random walk of the LSBs of DCT coefficients, skipping coefficients whose magnitudes are zero or unity. In the second pass, corrections are made to the magnitudes of the coefficients in order to ensure that the histogram of the DCTs of the stego image match those of the cover image.

Prior to embedding, OutGuess calculates the maximum length of a randomly spread message that can be embedded in the image during the first pass, while ensuring that one will be able to make corrections to adjust the histogram to its original values during the second pass. Let n_{01} be the number of all DCT coefficients whose magnitudes are not equal to 0 or 1. Because the majority of DCT coefficients are zeros, modifying them would introduce a large visible distortion. This is why OutGuess does not embed in zeros. Since 0 is paired with 1 in their LSB pair, coefficients equal to 1 are also skipped. After some thought, it is clear that the maximal correctable message length is determined by the frequencies of the most unbalanced LSB pair. Let us denote the histogram of DCT coefficients as $\mathbf{T}_c$. Due to the shape of the DCT histogram, the most unbalanced LSB pair is the pair $(-2, -1)$ (remember that the pair $(0, 1)$ is not used). Because $\mathbf{T}_c[-2] < \mathbf{T}_c[-1]$, after embedding the maximum correctable message we will need to move all remaining coefficients with value -2 to -1 in the second phase.

To obtain the maximum *correctable* relative embedding capacity, q, we start with the expression for the expected value of the histogram of DCT coefficients after embedding. Because the embedding mechanism is LSB embedding, we can use Equation 13.1, in Chapter 13 (p. 478), with qn_{01} as the length of the embedded message. The number of unused DCT coefficients with value -2 after embedding is $(1 - q)\mathbf{T}_c[-2]$, and this must be larger than the decrease in the number of coefficients with value -1, which is $\frac{q}{2}\mathbf{T}_c[-1] - \frac{q}{2}\mathbf{T}_c[-2]$, where the first term is the number of DCT coefficients with value -1 that are changed *from* the value -1 and the second term is the number of DCT coefficients that are altered *to* the value -1. Thus, we obtain the following inequality and eventually an upper bound on the correctable message length q:

$$(1 - q)\mathbf{T}_c[-2] \geq \frac{q}{2}\mathbf{T}_c[-1] - \frac{q}{2}\mathbf{T}_c[-2]$$

$$q \leq \frac{2\mathbf{T}_c[-2]}{\mathbf{T}_c[-1] + \mathbf{T}_c[-2]}.$$

This condition guarantees that on average there will be enough unused coefficients with magnitude -2 that can be flipped back to -1 to make sure that the frequencies of the LSB pair $(-2, -1)$ are preserved after embedding. Since this pair is the most unbalanced one, virtually all other pairs can be preserved using the same correction step, as well. We note that some very sparsely populated histogram bins in the tails of the DCT histogram may not be adjusted correctly

during the second phase, but since their numbers are statistically insignificant, the impact on statistical detectability is negligible.

Other approaches to histogram-preserving steganography for JPEG imagery can be found in [123, 193, 311, 381, 413].

Because neighboring pixels in the spatial domain are much more correlated than DCT coefficients in a JPEG file, steganographic schemes that embed messages in the spatial domain must preserve more than the histogram of pixels and thus require more complex models. Because first-order statistics cannot capture the correlations among neighboring pixels, the joint statistics of neighboring pixel *pairs* [136] or Markov chains [373, 394] may be used instead.

12.4.2 Model-Based Steganography

Model-based steganography is a general framework for constructing steganographic systems that preserve a chosen *model* of the cover Works [357] rather than its statistics. The cover Work, c, is modeled as a random variable that can be split into two components, $(\mathsf{c}_{inv}, \mathsf{c}_{emb})$, where c_{inv} is invariant to embedding and c_{emb} may be modified during embedding. For example when using LSB embedding, c_{inv} and c_{emb} correspond to the 7 most significant bits and the least significant bit of DCT coefficients, respectively.

The designer of the stego system first chooses a model for cover Works, which will be represented by conditional probabilities $P(\mathsf{c}_{emb}|\mathsf{c}_{inv})$. It is important that these probabilities can be calculated from the invariant portion only so that the same information is available to both the sender and the recipient. For each value $\mathbf{c}_{inv}$, let $\mathcal{L}(\mathbf{c}_{inv})$ be the set of pixels (or DCT coefficients) with their invariant part equal to $\mathbf{c}_{inv}$. The uniformly distributed message bits are run through an entropy decoder (e.g., an arithmetic decompressor with 0s and 1s occurring with probabilities $P(0|\mathsf{c}_{inv})$ and $P(1|\mathsf{c}_{inv})$) to change the message's uniform distribution to the required distribution expressed by the conditional probabilities. Thus, by replacing $\mathbf{c}_{emb}$ for pixels/DCT coefficients in $\mathcal{L}(\mathbf{c}_{inv})$ with the transformed message bits, we obtain a stego Work with similar statistical properties. The embedding and extraction procedures are illustrated in Figures 12.3 and 12.4.

The fraction of bits we can embed in each element of $\mathcal{L}(\mathbf{c}_{inv})$ is the entropy of $P(\mathsf{c}_{emb}|\mathsf{c}_{inv} = \mathbf{c}_{inv})$ or

$$H(P(\mathsf{c}_{emb}|\mathsf{c}_{inv} = \mathbf{c}_{inv})) = -\sum_{\mathbf{c}_{emb}} P_{\mathsf{c}_{emb}|\mathsf{c}_{inv}}(\mathbf{c}_{emb}|\mathbf{c}_{inv}) \log_2 P_{\mathsf{c}_{emb}|\mathsf{c}_{inv}}(\mathbf{c}_{emb}|\mathbf{c}_{inv}). \tag{12.7}$$

Thus, the total embedding capacity is

$$\sum_{\mathbf{c}_{inv}} |\mathcal{L}(\mathbf{c}_{inv})| H(P(\mathsf{c}_{emb}|\mathsf{c}_{inv} = \mathbf{c}_{inv})). \tag{12.8}$$

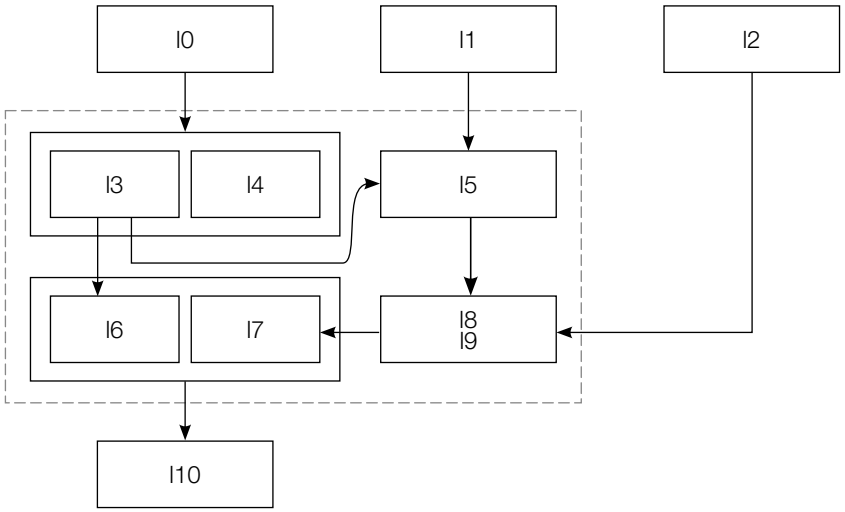

FIGURE 12.3

Embedding scheme for model-based steganography.

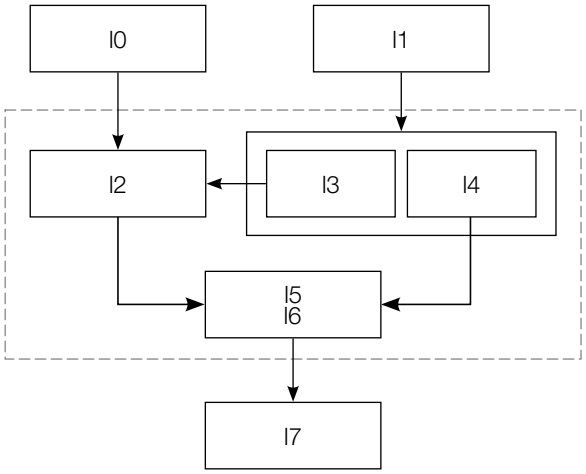

FIGURE 12.4

Extracting algorithm for model-based steganography.

To illustrate the model-based approach, let us consider a specific realization of this approach for JPEG images [357]. As with OutGuess, DCT coefficients with magnitudes of 0 or 1 are not used for embedding. The cover JPEG file is decomposed into 64 subsets corresponding to the 64 DCT frequencies. Let

us examine a specific DCT frequency, for example (2, 2), which corresponds to the fifth DCT coefficient in the zig-zag scan. Let $\mathbf{T}_c[i]$ be the histogram of all (2, 2) DCT coefficients in the cover image. The embedding mechanism will again be LSB embedding. Because LSB embedding preserves the sums $\mathbf{S}[2i] = \mathbf{T}_c[2i] + \mathbf{T}_c[2i+1]$ for all i, we can use this invariant and fit a parametric model through the points $(2i, \mathbf{S}[2i])$. For example, we can model the DCT coefficients using the generalized Cauchy model with pdf

$$\frac{p-1}{2s}\left|x/s + 1\right|^{-p},$$

and use a maximum likelihood estimator to determine the two model parameters, $p > 1$ and $s > 0$. By sampling this model at integer values, we can read out the expected values of the histogram at $\mathbf{T}_c[2i]$ and $\mathbf{T}_c[2i+1]$ for each LSB pair. Denoting the 7 most significant bits of an integer, a, as $MSB_7(a)$, we have

$$\begin{aligned} P\left(\mathbf{c}_{emb} = 0 | \mathbf{c}_{inv} = MSB_7(2i)\right) &= \frac{\mathbf{T}_c[2i]}{\mathbf{T}_c[2i] + \mathbf{T}_c[2i+1]} \\ &= 1 - P\left(\mathbf{c}_{emb} = 1 | \mathbf{c}_{inv} = MSB_7(2i)\right). \end{aligned} \tag{12.9}$$

The sender now collects all $\mathbf{S}[2i]$ DCT coefficients for the selected DCT frequency that are equal to $2i$ or $2i+1$. Their LSBs are replaced with a segment of the message that was decompressed using an arithmetic decompressor to the length $\mathbf{S}[2i]$. The purpose of the decompressor is to change a uniformly distributed bitstream to a biased bitstream according to the probabilities given by Equation 12.9.

The recipient repeats the same steps that the sender followed, building a model for each DCT coefficient, and obtaining the probabilities given by Equation 12.9. Then, the bits are extracted from the LSBs and passed through an arithmetic compressor to obtain the secret message. This process is shown in Figure 12.4.

This example of model-based steganography preserves the *model* of the histograms of all individual DCT coefficients while achieving a relatively large embedding capacity (0.8 bits per nonzero DCT coefficient for 80% quality JPEG images).

We now calculate the average number of bits embedded per unit distortion for this algorithm (its embedding efficiency). Let $p_0 = P\left(\mathrm{c}_{emb} = 0 \,|\, \mathrm{c}_{inv} = MSB_7(2i)\right)$. The number of bits embedded at each DCT coefficient whose 7 MSBs are equal to $2i$ is $H(p_0) = -p_0 \log_2 p_0 - (1-p_0)\log_2(1-p_0)$. The probability of an embedding change is the probability that the LSB is different than the embedded message bit. Because the message bits are prebiased to have the probability of zero, p_0, we

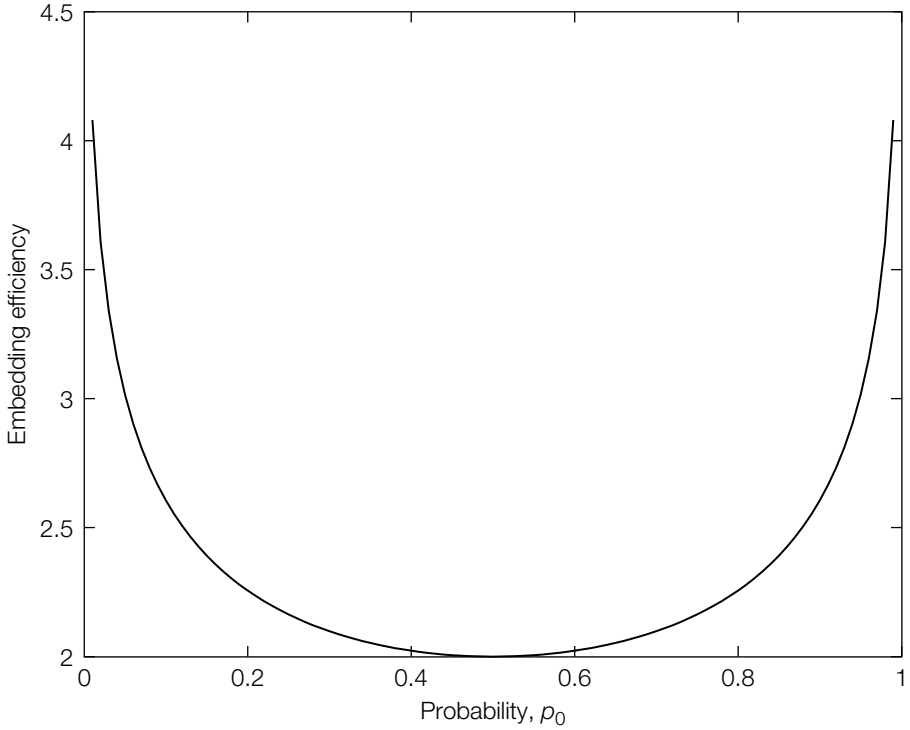

FIGURE 12.5

Embedding efficiency of model-based steganography.

have for the probability of change $p_0(1-p_0)+(1-p_0)p_0 = 2p_0(1-p_0)$. Thus, the embedding efficiency when embedding in coefficients with $MSB_7(2i)$ is

$$\frac{-p_0 \log_2 p_0 - (1-p_0)\log_2(1-p_0)}{2p_0(1-p_0)}. \tag{12.10}$$

The embedding efficiency is displayed in Figure 12.5. Note that is always greater than or equal to 2. This should be compared with embedding based on simple replacement of the LSBs. The latter's embedding efficiency is always equal to 2 because, on average, every other LSB is modified. In general, steganographic schemes with low embedding efficiency appear to be more statistically detectable than schemes with higher embedding efficiency. More detailed discussion on this topic appears in Section 12.5.1.

A more advanced version of this algorithm that preserves a specific second-order statistic was proposed in Sallee [357].

12.4.3 **Masking Embedding as Natural Processing**

In the previous sections, we discussed approaches to constructing practical steganographic schemes that strive to preserve some vital statistics (or a model) of Works. The advantage of such schemes is that they are secure in Cachin's sense as long as the statistics or model completely characterize the space of all Works. With the advancement of statistical image modeling, they also provide clues for building future schemes. Unfortunately, the job of the warden (steganalyst) is significantly simpler. Unlike Alice, the warden does not need to perfect her model of cover Works. All she needs to do is to find a deviation of Alice's model from reality. That could be a single statistical quantity that is not preserved by the embedder. It turns out that for each particular steganographic method, it is not that difficult to identify such a quantity. We give some examples in Section 13.2 on steganalysis. Unfortunately for Alice, while it is feasible to construct a steganographic scheme that preseves a specific model or statistics, it is much more complicated to preserve potentially hundreds of quantities used in current blind steganalysis methods.

This complication justifies investigation of alternative approaches to steganography, perhaps based on some heuristic principles. One possibility is to attempt to mask embedding as a natural process. Presumably, if the effect of embedding mimicked some natural process, it should be hard to argue that the Work was embedded rather than processed. In this section, we take a look at two practical realizations of this principle—one for images in the spatial domain and one for the JPEG format.

Stochastic Modulation

There are many noise sources that affect the acquisition of digital images. They include the shot noise due to quantum properties of light, dark current (the signal produced by the sensor when it is not exposed to light), circuit noise, readout noise, pixel-to-pixel nonuniformity (also known as pattern noise), quantization noise, etc. [181, 196, 207]. These stochastic components depend on ambient temperature and exposure time, or are intrinsic properties of the imaging sensor and the associated hardware.

The idea of stochastic modulation is to masquerade the embedding changes as a device noise. This is motivated by the plausible assumption that it may be difficult to determine whether the slightly increased noise level of the cover image is due to the presence of a hidden message or simply to environmental factors or different imaging hardware.

Let us assume that we want the impact of embedding to be the same as adding to the cover a noise signal with a given probability density function. Let $\mu[i]$ be an i.i.d. noise with pdf f_μ, and let $\mathbf{u}[i] = round(\mu[i])$

be the same sequence rounded to integers.[4] Thus, for any integers, k and i,

$$P(\mathbf{u}[i] = k) = \int_{k-0.5}^{k+0.5} f_{\mu}(x)dx. \tag{12.11}$$

The rounded noise sequence, $\mathbf{u}[i]$, is called the stego noise. One possibility to realize the embedding would be to use spread spectrum watermarking and embed 1 bit by adding to the cover image a spreading sequence modulated with the message bit. In order to reliably extract the message bits, however, the spreading sequence would have to be sufficiently long. This is not desirable for two reasons: we would impose a severe limit on the embedding capacity and also hide with a very low embedding efficiency (large distortion per embedded bit).

Before introducing stochastic modulation, we map the message bits to the interval $\{-1, 1\}$, rather than $\{0, 1\}$. We define a bit-assignment function, β, with the following property:

$$\beta(a + b, b) = -\beta(a, b) \tag{12.12}$$

for all integers a and b. The following function, for example, satisfies this requirement:

$$\begin{aligned} &\beta(a, b) = (-1)^a \quad \text{for} \quad 2i|b| \le a \le 2i|b| + |b| - 1, \\ &\beta(a, b) = -(-1)^a \quad \text{for} \quad (2i - 1)|b| \le a \le 2i|b| - 1 \\ &\beta(a, b) = 0 \text{ for } b = 0. \end{aligned} \tag{12.13}$$

The data embedding proceeds in the following manner. Using the stego key, the sender generates a pseudo-random path through the image and two independent stego noise sequences, $\mathbf{u}[i]$ and $\mathbf{v}[i]$, with the same probability mass function of Equation 12.11. The sender embeds the message bit, m, at the ith pixel, $\mathbf{c}[i]$, if and only if $\mathbf{u}[i] \neq \mathbf{v}[i]$. In this case, if $\beta(\mathbf{c}[i] + \mathbf{v}[i], \mathbf{u}[i] - \mathbf{v}[i]) = m$, the sender replaces $\mathbf{c}[i]$ with $\mathbf{s}[i] = \mathbf{c}[i] + \mathbf{v}[i]$. If $\beta(\mathbf{c}[i] + \mathbf{v}[i], \mathbf{u}[i] - \mathbf{v}[i]) = -m$, then due to the antisymmetry of Equation 12.12, we must have $\beta(\mathbf{c}[i] + \mathbf{u}[i], \mathbf{u}[i] - \mathbf{v}[i]) = -\beta(\mathbf{c}[i] + \mathbf{v}[i] + \mathbf{u}[i] - \mathbf{v}[i], \mathbf{u}[i] - \mathbf{v}[i]) = m$, and thus, we replace $\mathbf{c}[i]$ with $\mathbf{s}[i] = \mathbf{c}[i] + \mathbf{u}[i]$. If $\mathbf{u}[i] = \mathbf{v}[i]$, we replace $\mathbf{c}[i]$ with $\mathbf{s}[i] = \mathbf{c}[i] + \mathbf{u}[i]$ and reembed the same bit m at the next pixel along the pseudo-random walk.

The sender generates two independent stego noise sequences with the required probability mass function and then adds one or the other to embed the required bit at each pixel. If the two stego noise samples are the same, the embedding process cannot embed a bit, in which case the sender still adds

[4] Here, we again assume that the dynamic range of the cover Work is $0, \ldots, 255$.

the noise to the image but reembeds the same bit at the next pixel. Also, because images have finite dynamic range (e.g., the pixel values of a grayscale image are between 0 and 255), care must be taken when the embedding value, $\mathbf{s}[i] = \mathbf{c}[i] + \mathbf{v}[i]$, goes out of range. In this case, we must deviate from the stego noise model and instead add $\mathbf{v}'[i]$, such that $\beta(\mathbf{c}[i] + \mathbf{v}'[i], \mathbf{u}[i] - \mathbf{v}[i]) = m$ with $|\mathbf{v}'[i] - \mathbf{v}[i]|$ as small as possible.

To extract the message, the recipient first uses his or her stego key to generate both sequences, $\mathbf{u}[i]$ and $\mathbf{v}[i]$, and then reads the message as $\beta(\mathbf{s}[i], \mathbf{u}[i] - \mathbf{v}[i] \,|)$ skipping the cases when $\mathbf{u}[i] = \mathbf{v}[i]$.

INVESTIGATION

Embedding Capacity of Stochastic Modulation

The embedding capacity of stochastic modulation is $1 - P(\mathbf{u} = \mathbf{v}) = 1 - \sum_k [P(\mathbf{u}[i] = k)]^2$ bits per pixel and can be evaluated using Equation 12.11. For example, if μ is Gaussian $N(0, \sigma^2)$, the relative embedding capacity is shown in Figure 12.6. Note that by adding quantized Gaussian noise $N(0,1)$ the sender can be communicating at a little over 0.7 bits per pixel.

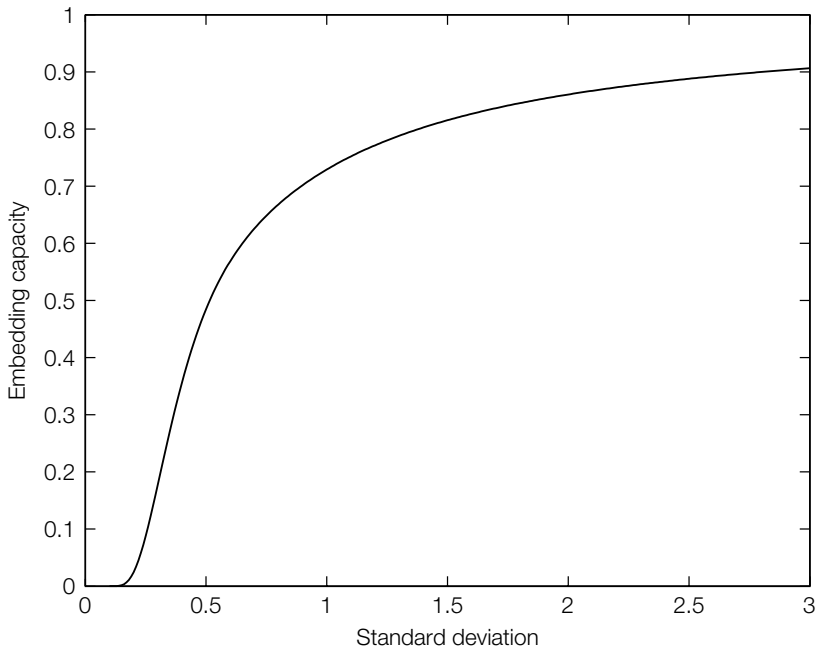

FIGURE 12.6

Embedding capacity of stochastic modulation.

It should not be surprising that the capacity is a function of the standard deviation, σ, because the noise sequences when rounded to integers are more likely to be equal for small σ.

Although stochastic modulation is much less detectable than LSB embedding, modern steganalysis tools based on feature extraction and classification (see Section 13.2.4 on blind steganalysis) are capable of detecting it relatively reliably, at least for payloads above 0.1 bpp. This is because the processing that occurs in digital cameras or scanners contains color interpolation and various filtering operations that create many complex dependencies among neighboring pixels. Injecting noise after all this processing disturbs these dependencies and enables relatively reliable steganalysis.

The F5 Embedding Algorithm

The F5 algorithm [442] was proposed with the goal to develop concepts and a practical embedding method for JPEG images that would provide high steganographic capacity without sacrificing security. Instead of LSB flipping, the embedding operation in F5 decrements the absolute value of the DCT coefficient by one. This operation preserves the natural shape of the DCT histogram, which looks after embedding as if the cover image was originally compressed using a lower quality factor.

The F5 algorithm embeds message bits along a pseudo-random path determined from a user pass-phrase. The DC terms and zero magnitude coefficients are skipped and not used for embedding.

When a coefficient's magnitude is changed from either 1 or −1 to zero, it is called "shrinkage." If shrinkage occurs, we have to reembed the same bit, which is always a 0, at the next coefficient. This is because the recipient will read the message bits from AC coefficients that are not equal to zero. However, if we reembed 0 bits, we will embed a *biased* bitstream with more zeros than ones. This would cause "staircase" artifacts in the histogram because of its monotonicity on $(-\infty, 0]$ and $[0, \infty)$. F5 solves this problem by redefining the LSB for negative numbers: $LSB(x) = 1 - x \bmod 2$ for $x < 0$ and $LSB(x) = x \,\%\, 2$ otherwise. This solves the problem because now shrinkage occurs when embedding both 0s and 1s with approximately equal probability. This is due to the symmetrical shape of the DCT histogram.

The embedding capacity of F5 is $n - \mathbf{T}_c[0] - n/64 - (\mathbf{T}_c[-1] + \mathbf{T}_c[1])/2$, where n is the total number of DCT coefficients and $\mathbf{T}_c$ is the histogram of all DCT coefficients. The first three terms $n - \mathbf{T}_c[0] - n/64$ give the number of all nonzero AC coefficients, while the last term is the loss due to shrinkage. F5 is a relatively high-capacity algorithm that, on average, enables embedding

at 0.75 bits per nonzero DCT coefficient for JPEG images compressed with a quality factor of 80. As an additional improvement, for short messages F5 employs matrix embedding that increases the embedding efficiency. This general method for minimizing the number of embedding changes is explained in Section 12.5.1.

While the F5 algorithm modifies the histogram of DCT coefficients, some crucial characteristics of the histogram are preserved, such as its monotonicity and monotonicity of increments. The F5 algorithm cannot be detected using the histogram attack described in Section 13.2.1, because the embedding is not based on exchanging any fixed pairs of values. It is, however, vulnerable to attacks that use a process called calibration (see Section 13.2.3).

12.5 MINIMIZING THE EMBEDDING IMPACT

Cachin's definition of steganographic security calls for the Kullback-Leibler distance between the probability distributions of cover and stego Works to be as small as possible. Intuitively, we can try to achieve this by minimizing the distortion between the cover and stego Works and by restricting distortions to portions of the cover Work that are difficult to model. By so doing, we make it harder for the warden to collect evidence that steganography is taking place. This section addresses both issues (i.e., minimizing the embedding distortion and adaptive embedding).

We start by casting the problem of data hiding within the framework of coding theory. This enables us to derive bounds on the maximum number of secret message bits that can be embedded in a cover Work given a limited number of embedding changes. Remember that for LSB embedding, on average we embed 2 bits for each embedding change. This is because half the time, the LSB value is the same as the message bit and therefore no change is needed. In fact, we can do much better than this if the message is shorter than the embedding capacity. Section 12.5.1 discusses this in detail.

Section 12.5.2 then discusses methods by which the embedder can adaptively determine the location of the alterations. A shared selection rule is no longer required, that is, Bob does not need to know the location of the changes in order to decode the message. The advantage of this is that Alice can now utilize side information only available to her in order to minimize the impact of embedding. For example, assuming she has access to *unquantized* DCT coefficients while JPEG compressing the cover Work, she can confine her embedding changes to those coefficients that experience the smallest *combined* distortion due to quantization and embedding.

12.5.1 Matrix Embedding

The embedding changes can encode the message bits in many different ways. For example, in LSB embedding, the LSBs are changed to match the secret message bits (e.g., the extracted message is simply the string of LSBs of individual pixels). This way, we always embed, on average, 2 bits per one embedding change. However, we can do substantially better if we adopt a more clever embedding scheme. In particular, if the message is shorter than the embedding capacity, one can, for example, use the position of changes to encode more bits per one change. Let us take a look at the following simple example.

Let us assume that we have a group of three pixels with grayscale values x_1, x_2, x_3, and we wish to embed 2 message bits, b_1, b_2. It seems that a reasonable approach might be to simply embed b_1 at x_1 and b_2 at x_2, modifying the LSB of the pixels to match the corresponding message bits. Assuming the 2 bits are 0 or 1 with equal probability, the expected number of changes to the whole group of pixels to embed both bits is 1. Thus, we embed at embedding efficiency of 2 or 2 bits per one change. However, we can do better. Let us encode $b_1 = LSB(x_1)$ XOR $LSB(x_2)$ and $b_2 = LSB(x_2)$ XOR $LSB(x_3)$. If the values of the cover Work satisfy both equations with equality, no embedding changes are necessary. If the first one is satisfied but not the second one, we simply flip the LSB of x_3. If the second one is satisfied but not the first one, we flip the LSB of x_1. If neither one is satisfied, we flip x_2. Because all four cases are equally likely with probability 1/4, the expected number of changes is 3/4, which is less than what we had before. This simple trick is a special case of a much more general principle called *matrix embedding*, [40, 159, 419], which we now describe in more detail.

We will assume that the individual elements of the cover Work are represented as bits via some bit-assignment function, β (e.g., the LSB). The exact mechanism of the embedding operation is not essential to us now. We only need to know that the operation changes the bit assigned to the element.

Let us assume that we have a pair of embedding and extraction functions such that we can embed any message $\mathbf{m} \in \mathcal{M}$ using, at most, R changes

$$Emb\colon \{0,1\}^n \mathcal{M} \to \{0,1\}^n \text{ and } Ext\colon \{0,1\}^n \to \mathcal{M}, \tag{12.14}$$

such that for all $\mathbf{m} \in \mathcal{M}$ and $\mathbf{x} \in \{0,1\}^n$,

$$Ext(Emb(\mathbf{x}, \mathbf{m})) = \mathbf{m}$$
$$d(\mathbf{x}, Emb(\mathbf{x}, \mathbf{m})) \leq R.$$

As defined in Section 12.2, the embedding capacity in bits is $\log_2 |\mathcal{M}|$. If R_a is the expected number of embedding changes over uniformly distributed cover Works, $\mathbf{x}$, and messages, $\mathbf{m}$, then $R_a \leq R$, and we define the *embedding efficiency*, of this scheme as $e = \frac{\log_2 |\mathcal{M}|}{R_a}$. We also define the *lower embedding*

efficiency, $\underline{e} = \frac{\log_2 |\mathcal{M}|}{R}$, which is always less than or equal to the embedding efficiency, e, since $R_a \leq R$.

Matrix embedding is a form of syndrome coding. To define a syndrome, remember that for an $[n, k]$ linear code, $\mathcal{C}$, k-bits are transmitted as an n-bit code word, where $n > k$. In error correcting codes, the k-bits are usually referred to as the message, but for reasons that will soon be apparent, we refer to them as the base bits. Each $[n, k]$ linear code, $\mathcal{C}$, is completely described by its generator matrix, $\mathbf{G}$, which is a regular binary matrix with k rows and n columns. All code words, $c \in \mathcal{C}$, are obtained as linear combinations of the rows of $\mathbf{G}$, where the coefficients in the linear combination are the k base bits.

The detector receives a possibly corrupted code word, $\mathbf{c}'$. The vector, $\mathbf{Hc}'$, is called the syndrome, where $\mathbf{H}$ is the $(n-k) \times n$ parity check matrix of the code. If $\mathbf{c}'$ is an element of $\mathcal{C}$ (i.e., is uncorrupted), then its syndrome is

$$\mathbf{Hc}' = 0. \tag{12.15}$$

If $\mathbf{c}'$ is corrupted, then

$$\mathbf{Hc}' \neq 0.$$

Note that the syndrome is an $(n-k)$ dimensional vector.

As discussed in Appendix A.2, there are many corrupted code words that map to the same syndrome, $\mathbf{s}$, and this set of corrupted code words forms a coset that we denote as $\mathcal{C}(\mathbf{s})$ (i.e., the set of $\mathbf{c}'$), such that $\mathbf{Hc}' = \mathbf{s}$.

Matrix embedding (and syndrome coding) hide the steganographic message as the syndrome. Thus, for an $[n, k]$ code, we can embed $(n-k)$ steganographic message bits. Since the message is encoded in the syndrome and *not* in the k base bits, as is the case for conventional coding for error correction, we refrain from calling the k base bits message bits.

To perform syndrome coding, let $\mathbf{m}$ denote our $(n-k)$-bit message (syndrome). To encode $\mathbf{m}$, first, let $\mathbf{x}$ denote the n-bits from the cover Work that will hide the message. We must alter $\mathbf{x}$ to $\mathbf{y}$ such that

$$\mathbf{Hy} = \mathbf{m}.$$

In other words, the recipient extracts the message from the stego Work $\mathbf{y}$ as its syndrome. Let $\mathbf{y} = \mathbf{x} + \mathbf{e}$, where $\mathbf{e}$ is the vector of embedding changes. The Hamming weight of $\mathbf{e}$ is equal to the number of embedding changes. Then, we have that

$$\mathbf{H}(\mathbf{x} + \mathbf{e}) = \mathbf{Hy} = \mathbf{m},$$

and thus,

$$\mathbf{He} = \mathbf{m} - \mathbf{Hx}. \tag{12.16}$$

The right-hand side of Equation 12.16 defines a syndrome coset,[5] $C(\mathbf{m} - \mathbf{Hx})$, and the solution to Equation 12.16 is that $\mathbf{e}$ is the coset leader, $\mathbf{e}_\mathbf{L}$, of $C(\mathbf{m} - \mathbf{Hx})$, that is, the element of the coset with the smallest Hamming weight (smallest number of embedding changes). And, since the Hamming weight of $\mathbf{e}_\mathbf{L}$ is less than or equal to the covering radius, R, of the code, then the number of embedding changes is also less than or equal to R. We can summarize by stating that a linear $[n, k]$ code, C, with covering radius, R, can be used to construct an embedding scheme capable of communicating $(n - k)$ bits using at most R changes.

Depending on the choice of the linear code, C, different matrix embedding schemes can be constructed. The simplest scheme that is, for instance, implemented in the steganographic algorithm F5 [442] is based on binary Hamming codes with $n = 2^p - 1$ and $k = 2^p - 1 - p$, where p is a positive integer. The parity check matrix, $\mathbf{H}$, of this code has dimensions $p \times (2^p - 1)$, and its columns are binary representations of the numbers $1, \ldots, 2^p - 1$. We can say that $\mathbf{H}$ has as its columns all possible nonzero syndromes. For example, the parity check matrix, $\mathbf{H}$, for a Hamming code with $p = 3$ is

$$H = \begin{pmatrix} 0\,0\,0\,1\,1\,1\,1 \\ 0\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,1\,0\,1 \end{pmatrix}.$$

Notice that any nonzero syndrome, $\mathbf{s}$, is a column in $\mathbf{H}$, say the ith column. The coset leader, $\mathbf{e}_\mathbf{L}$, of $C(\mathbf{s})$ is a vector with only a nonzero element in its ith position. Thus, all coset leaders of this code have Hamming weight 1 and a code covering radius $R = 1$.

To summarize, Hamming codes enable embedding of $n - k = p$ bits in $n = 2^p - 1$ pixels by making at most one change. The reader is encouraged to verify that the "trick" shown in the introduction of this section corresponds to matrix embedding using Hamming codes with $p = 2$ and parity check matrix

$$\mathbf{H} = \begin{pmatrix} 1\,1\,0 \\ 0\,1\,1 \end{pmatrix}.$$

INVESTIGATION

Matrix Embedding Using Hamming Codes

In this investigation, we explain how matrix embedding using Hamming codes can be implemented in practice. The benefit of using this coding scheme is

[5] Note that the sender knows the syndrome, $\mathbf{s}$, because $\mathbf{m}$ is the message to be communicated and $\mathbf{x}$ is the vector of cover Work bits.

demonstrated by showing the decrease in the average number of embedding changes needed to embed a message.

Let us assume that the sender wants to communicate K bits in a cover Work consisting of N pixels. This means that the relative message length is $\alpha = K/N$. Since matrix embedding using Hamming codes embeds p bits in $2^p - 1$ pixels, the sender will minimize the number of embedding changes by choosing the largest value of p that provides sufficient relative payload, that is,

$$\alpha_{p+1} = \frac{p+1}{2^{p+1}-1} < \alpha \leq \frac{p}{2^p - 1} = \alpha_p.$$

The sender starts by dividing the cover into $N/(2^p - 1)$ blocks, each consisting of $2^p - 1$ pixels. In each block, p message bits are embedded by performing at most one embedding change in the same manner as above, communicating p bits as a syndrome $\mathbf{m} = \mathbf{Hy}$, where $\mathbf{y}$ is the vector of bits from the stego image.

We now calculate the average number of embedding changes and the embedding efficiency of this matrix embedding scheme. Remember that the embedding efficiency is the average number of message bits embedded per unit distortion (one embedding change).

When embedding p pseudo-random bits, $\mathbf{m}$, in a pixel block, then, with probability $1/2^p$, the cover pixel block, $\mathbf{x}$, will already communicate the required message (i.e., $\mathbf{m} = \mathbf{Hx}$), and no embedding change will be necessary in this case. In all other cases, the sender makes exactly one embedding change with probability $1 - 1/2^p$. Thus, the average number of embedding changes to embed p bits is

$$0 \times 1/2^p + 1 \times (1 - 1/2^p) = 1 - /2^p,$$

and the embedding efficiency, e_p, is

$$e_p = \frac{p}{1 - 2^{-p}}, \tag{12.17}$$

Table 12.1 shows the relative message length and the embedding efficiency for various values of p. Note that with increasing p, the embedding efficiency increases while the relative payload decreases. Also, Hamming codes cannot be applied for embedding messages of relative length larger than 2/3.

Finally, note that the parameter, p, chosen by the sender depends on the message length and thus needs to be communicated to the recipient in the stego image itself. This can be arranged in a variety of ways, depending on the steganographic scheme. For example, the sender can reserve a few pixels in the image and embed information about p using LSB embedding.

Table 12.1 Relative message length and embedding efficiency for matrix embedding using binary Hamming codes $[2^p - 1, 2^p - 1 - p]$.

p	Relative message length, α_p	Embedding efficiency, e_p
1	1.000	2.000
2	0.667	2.667
3	0.429	3.429
4	0.267	4.267
5	0.161	5.161
6	0.093	6.093
7	0.055	7.055
8	0.031	8.031
9	0.018	9.018

Upper Bound on Embedding Efficiency

In the previous section, we saw how linear codes can be used to decrease the number of changes needed to embed a message using a procedure called matrix embedding. A natural question to ask is, what are the limits of this approach? For example, given a relative message length, α, what is the average minimal number of embedding changes needed to embed it? Alternatively, we may ask about the maximum relative message length embeddable using a given number of changes. Answering these questions is the subject of this section. The bounds derived here serve as a benchmark against which matrix embedding schemes can be evaluated.

The maximum number of messages that can be communicated by making at most R changes is bound by the total number of ways one can make R or fewer changes. This is because the sender and the receiver could share a lookup table where each embedding change would be associated with one message. Because there are $\sum_{i=0}^{R} \binom{n}{i} 2^i$ ways in which one can make R or fewer changes in n pixels, we obtain the following bound on the maximal number of bits that can be embedded:

$$\log_2 |\mathcal{M}| \leq \log_2 \sum_{i=0}^{R} \binom{n}{i} 2^i. \tag{12.18}$$

We now apply a well-known inequality from information theory [89]:

$$\log_2 \sum_{i=0}^{R} \binom{n}{i} 2^i \leq nH(R/n), \tag{12.19}$$

where $H(x) = -x\log_2 x - (1-x)\log_2(1-x)$ is the *binary entropy function* shown in Figure 12.7. Combining Equations 12.18 and 12.19, we obtain an upper bound on the maximal relative payload embeddable using R changes, that is

$$\alpha_{max} = \frac{\log_2 |\mathcal{M}|}{n} \leq H(R/n). \tag{12.20}$$

Sometimes, we may be interested in finding the maximal possible embedding efficiency *given* a certain relative payload α. This would answer the question of what is the minimal number of embedding changes that are, on average, needed to embed a given payload. This bound can be obtained by first rewriting

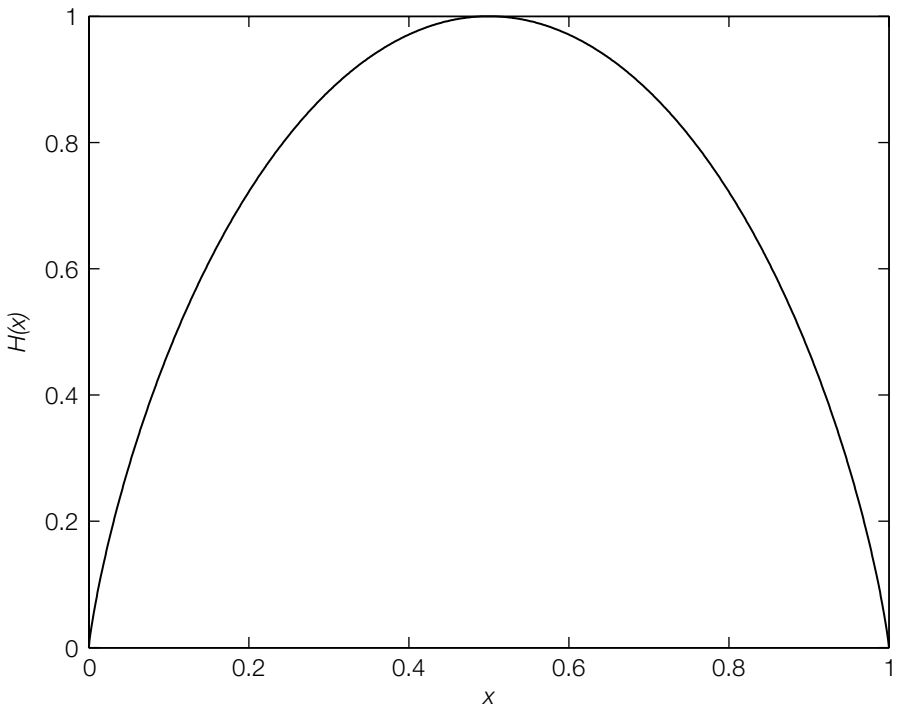

FIGURE 12.7

Binary entropy function.

Equation 12.20 as $n/R \leq 1/H^{-1}(\alpha)$, or $\log_2 |\mathcal{M}|/R \times n/\log_2 |\mathcal{M}| \leq 1/H^{-1}(\alpha)$, which gives

$$\underline{e} = \frac{\log_2 |\mathcal{M}|}{R} \leq \frac{\alpha}{H^{-1}(\alpha)}, \tag{12.21}$$

where the range of the inverse function H^{-1} is the interval $[0, 1/2]$. This last bound gives a useful bound on the lower embedding efficiency achievable using an arbitrary matrix embedding scheme that can embed a relative message length, α. We note that this bound is tight and can be reached with matrix embedding [159]. We plot the bound as a function of the relative message length, α, in Figure 12.8. The naive LSB embedding method always performs with an embedding efficiency $e = 2$ at any payload, α. Binary Hamming codes, also shown in the figure, provide substantial advantage, for example, giving an embedding efficiency as high as $e = 6$ for a relative message length $\alpha = 0.1$.

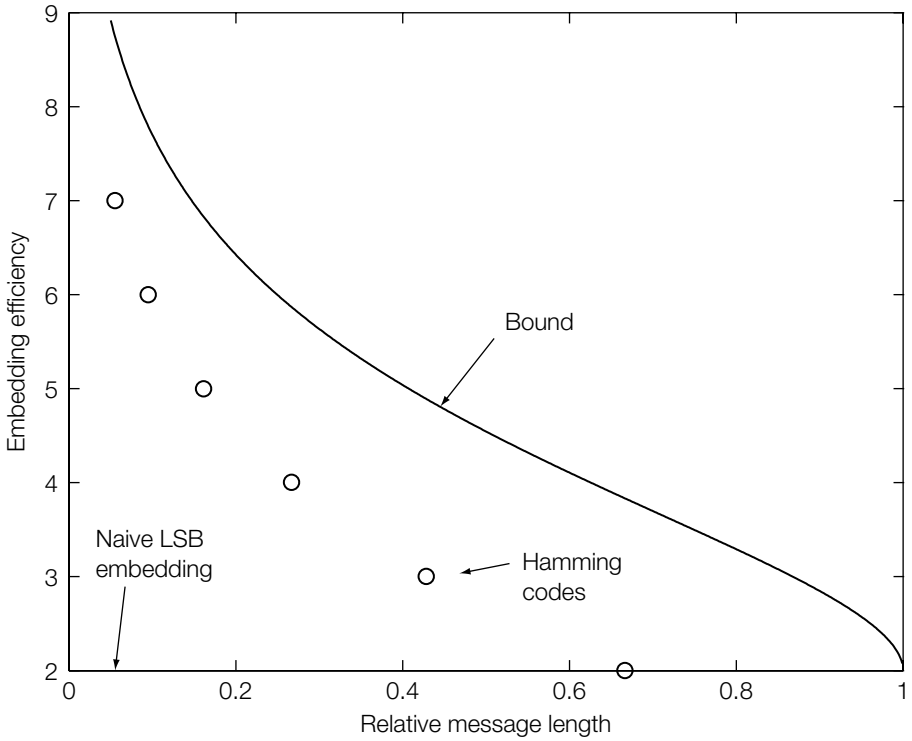

FIGURE 12.8

Embedding efficiency, e_p, in bits per embedding change as a function of relative message length, α, for matrix embedding using binary Hamming codes.

In general, it is fairly difficult to find codes with embedding efficiency close to the bound of Equation 12.21. Random linear codes with small codimension ($n - k$) have been shown to achieve slightly better performance than Hamming codes [149]. Nonlinear codes offer better embedding efficiency at the cost of a more complicated embedding [39]. Matrix embedding methods for large payloads were studied in Fridrich and Soukal [152].

12.5.2 Nonshared Selection Rule

In the previous section, we introduced matrix embedding as a means for decreasing the number of embedding changes and thus reducing the statistical impact of embedding a message into a cover Work. Besides making fewer embedding changes, the sender may attempt to restrict the embedding to those parts of the cover Work where it is intuitively more difficult to detect the changes. The rule used to select the placement of embedding changes is called the *selection rule*. If the selection rule is only known to the sender but not to the recipient, we speak of a *nonshared* selection rule.

To motivate the need for nonshared selection rules in steganography, imagine the following situation. The sender has a raw, never compressed image and wants to embed information in its JPEG compressed form. Can the knowledge of the raw image help us better conceal the embedding changes? Intuitively, it should. For example, when compressing the image, the sender can inspect the DCT coefficients after they are divided by the associated quantization steps but *before* they are rounded to integers, and select for embedding those coefficients whose fractional part is close to 0.5. Such coefficients experience the *largest* quantization error during JPEG compression and the *smallest combined error* (rounding + embedding) if rounded to the other value. When rounding the coefficient 5.47, for instance, we can embed a bit by rounding it to 5 or to 6. The rounding distortion (without embedding) is 0.47. If embedding requires rounding to 6, the combined rounding and embedding distortion is only slightly larger, 0.53. The obvious problem with this idea, however, is that the recipient will not be able to tell which DCT coefficients in the stego JPEG file were used for embedding, because it is not possible to completely undo the loss due to rounding by JPEG compression. This is an example of a situation in which the sender *cannot* share the selection rule with the recipient.

The problem of nonshared or partially shared selection rules occurs quite frequently in steganography. Consider, for example, adaptive steganography [71, 220] where the pixels in the cover Work are chosen using a selection rule based on their neighborhood. Because the act of embedding itself modifies the pixel values, then, depending on the selection rule, it is possible that the recipient will not recover the same set of message-carrying pixels as the sender. As an example, consider the following selection rule. The sender calculates for each pixel in the cover Work the variance of all pixels in its local 3×3 neighborhood and embeds m message bits using LSB embedding into the LSBs of m pixels

with the largest local variance. When the same selection rule is applied by the recipient of the stego image, it may well happen that the m pixels with the largest local variance will not be completely the same as those selected by the sender. This may prevent the recipient from extracting the message. It obviously would be very useful to have a general method that would enable construction of steganographic schemes with nonshared selection rules.

Communication using nonshared selection rules in steganography has been called "writing on wet paper" [150]. To explain this metaphor, imagine that the cover image, $\mathbf{x}$, was exposed to rain and the sender is only allowed to slightly modify the dry spots of $\mathbf{x}$ (the selected pixels) but not the wet spots. During transmission, the stego image, $\mathbf{y}$, dries out and thus the recipient does not know which pixels the sender used (the recipient has no information about the dry pixels). This is equivalent to a well-known problem in information theory called *writing in memory with defective cells* [119, 182, 412, 466]. Here, a computer memory contains n cells, out of which $(n - k)$ cells are permanently stuck at either 0 or 1. The device that writes data into the memory knows the locations and status of the stuck cells. The task is to write as many bits as possible into the memory (up to k) so that the reading device, which does not have any information about the stuck cells, can correctly read the data. Clearly, writing on wet paper is formally equivalent to writing in memories with stuck cells (stuck cells = wet pixels).

The defective memory is a special case of the Gel'fand-Pinsker channel with an informed sender [161] that we already encountered in Chapter 5. It turns out that syndrome coding, which we discussed in the previous section on matrix embedding, is an indispensible tool for solving the problem of nonshared selection rules.

Wet Paper Codes with Syndrome Coding

In this section, we present a coding method by which it is possible to construct steganographic schemes with nonshared selection rules. The method is similar to matrix embedding as it is also based on syndrome codes (i.e., the message is communicated as a syndrome of some linear code).

We again assume that the Work is represented via some bit-assigment function, as a vector of bits, $\mathbf{x}$, for example, the LSBs. The sender uses some selection rule to choose k changeable pixels $\mathbf{x}[j]$, $j \in \mathcal{J} \subset \{1, \ldots, n\}$, and $|\mathcal{J}| = k$. Using the writing on wet paper metaphor, the set of pixel indices $\mathcal{J}$ indicate "dry" pixels that may be modified by the sender. The remaining pixels, corresponding to "wet" pixels, are not to be modified during embedding.

Following the mechanism of syndrome codes, to communicate m bits $\mathbf{m} \in \{0, 1\}^m$, the sender modifies some changeable pixels in the cover Work so that the stego Work, $\mathbf{y}$, satisfies

$$\mathbf{D}\mathbf{y} = \mathbf{m}, \tag{12.22}$$

where $\mathbf{D}$ is an $m \times n$ parity check matrix shared by the sender and the recipient. This embedding mechanism is almost identical to matrix embedding with one important difference: The sender now cannot modify all pixels but only the changeable ones. The recipient, again, extracts the message by calculating the syndrome, $\mathbf{Dy}$, from the stego Work, $\mathbf{y}$.

The embedding requires solving the system of linear equations of Equation 12.22, where the unknowns are the bits, $\mathbf{y}[j], j \in \mathcal{J}$, and the remaining bits of $\mathbf{y}$ are known. In order to better see what kind of task the sender has to perform, we rewrite Equation 12.22 as

$$\mathbf{Dv} = \mathbf{m} - \mathbf{Dx} \tag{12.23}$$

using the variable $\mathbf{v} = \mathbf{y} - \mathbf{x}$. The nonzero elements of $\mathbf{y}$ correspond to the pixels the sender must change to satisfy Equation 12.22. In Equation 12.23, there are k unknowns, $\mathbf{v}[j], j \in \mathcal{J}$, corresponding to changeable pixels, while the remaining $(n - k)$ values, $\mathbf{v}[i], i \notin \mathcal{J}$, are known and must be equal to zero. Thus, on the left-hand side, the sender can remove from $\mathbf{D}$ all $(n - k)$ columns corresponding to indices $i \notin \mathcal{J}$ and also remove from $\mathbf{v}$ all $(n - k)$ elements $\mathbf{v}[i]$ with $i \notin \mathcal{J}$. Keeping the same symbol for the pruned vector, $\mathbf{v}$, Equation 12.23 becomes

$$\mathbf{Hv} = \mathbf{z}, \tag{12.24}$$

where $\mathbf{H}$ is an $m \times k$ matrix consisting of those k columns of $\mathbf{D}$ corresponding to indices $\mathcal{J}$, $\mathbf{v}$ is an unknown $k \times 1$ vector holding the embedding changes, and $\mathbf{z} = \mathbf{m} - \mathbf{Dx}$ is a known $m \times 1$ right-hand side. At this point, the problem becomes *equivalent* to matrix embedding with a parity check matrix $\mathbf{H}$. Finding any member, $\mathbf{v}$, of the coset, $\mathcal{C}(\mathbf{z})$, amounts to successfully embedding the message. Choosing the coset leader minimizes the number of embedding changes.

When we carefully compare this task with matrix embedding, we discover one important difference. While in matrix embedding, we could impose a specific structure on the matrix, $\mathbf{H}$, such as requiring it to be the parity check matrix of Hamming codes; here, $\mathbf{H}$ is obtained as a submatrix of a larger matrix, $\mathbf{D}$, via a selection process dictated by the cover Work or by some side information. Thus, when embedding into two cover Works of the same dimensions, the two submatrices will, in general, be different. It is not easy to impose a structure on $\mathbf{H}$ that would enable us to use some known codes.

One strategy the sender may accept is to simply solve Equation 12.24 using Gaussian elimination. However, the complexity of Gaussian elimination is cubic in the number of equations, k, which is the number of changeable pixels. Thus, using Gaussian elimination to embed large payloads would not be feasible. Another possibility is to use random, sparse matrices, $\mathbf{D}$, for which the task of solving the linear system can become very easy. This idea is explored in the next section.

Nonshared Selection Rules Using the Matrix LT Process

In the previous section, we explained how nonshared selection rules can be realized in steganography using syndrome coding and a parity check matrix, **D**, shared between the sender and the recipient. In this section, we show how solving the linear system of Equation 12.24 can be implemented very efficiently, by choosing **D** from a special class of random, sparse matrices.

To explain the basic idea, imagine that in the linear system $\mathbf{Hv} = \mathbf{z}$ that the sender needs to solve, there exists a column with exactly one 1 in, say, the $j(1)$th row. The sender swaps this column with the first column and then swaps the first and $j(1)$th rows to bring the 1 to the upper left corner of **H**. Now we will apply the same step again while ignoring the first column and row of the permuted matrix. We again find a column with only one 1, say, in the $j(2)$th row[6] and swap the column with the second column of **H** followed by swapping the second and $j(2)$th rows. As a result, we will obtain a matrix with one's on the first two elements of its main diagonal and zeros below them. We can continue this process, this time ignoring the first two columns and rows. If we are lucky, this process of row and column swapping will eventually produce a permuted matrix with an upper-diagonal form. Such a system can be efficiently solved using standard back substitution, as in Gaussian elimination. We call this permutation procedure the *matrix LT process* because it was invented in the theory of erasure correcting codes called LT codes [275]. Note that if, at some step during the permutation process, we cannot find a column with exactly one 1, then the matrix LT processed failed.

The big question is what properties should the matrix **D** have in order for the permutation process to successfully finish? It turns out that as long as the Hamming weights of columns of **D** (and thus of **H**, too) follow a certain probability distribution, then, with high probability, the matrix LT process will succeed. This distribution is known as a robust soliton distribution (RSD) [275] and is defined as follows.

The probability that a column in **D** has Hamming weight i, $1 \leq i \leq k$, is $\frac{\rho[i] + \tau[i]}{\eta}$, where

$$\begin{aligned}
\rho[i] &= \tfrac{1}{m} \quad i = 1 \\
&= \tfrac{1}{i(i-1)} \quad i = 2, \ldots, m, \\
\tau[i] &= \tfrac{R}{im} \quad i = 1, \ldots, m/R - 1 \quad \text{and} \\
&= \tfrac{R \ln(R/\delta)}{m} \quad i = m/R \\
&= 0 \quad i = m/R + 1, \ldots, m,
\end{aligned}$$

[6] Since we are ignoring the first row, this column may have another 1 as its first element.

$$\eta = \sum_{i=1}^{m} (\rho[i] + \tau[i]) \tag{12.25}$$

and

$$R = c \ln(m/\delta)\sqrt{m} \tag{12.26}$$

for some suitably chosen constants δ and c. The choice of these constants will be discussed later.

An example of an RSD distribution for $k = 1{,}000$, and $c = 0.1$ is shown in Figure 12.9. By inspecting the figure, we can get a feeling for why this distribution looks the way it does. First, note that the distribution prefers columns of low Hamming weight while denser columns are less frequent. This intuitively guarantees that the LT process will always find a column with just one 1. The purpose of the mysterious spike at Hamming weight 27 is to make sure that the matrix will be regular with high probability. Without the spike, the matrix would become too sparse and not be of full rank.

To generate a matrix where the number of ones in its columns follows an RSD distribution, we first generate a sequence of integers, $w_1, w_2, \ldots,$ that

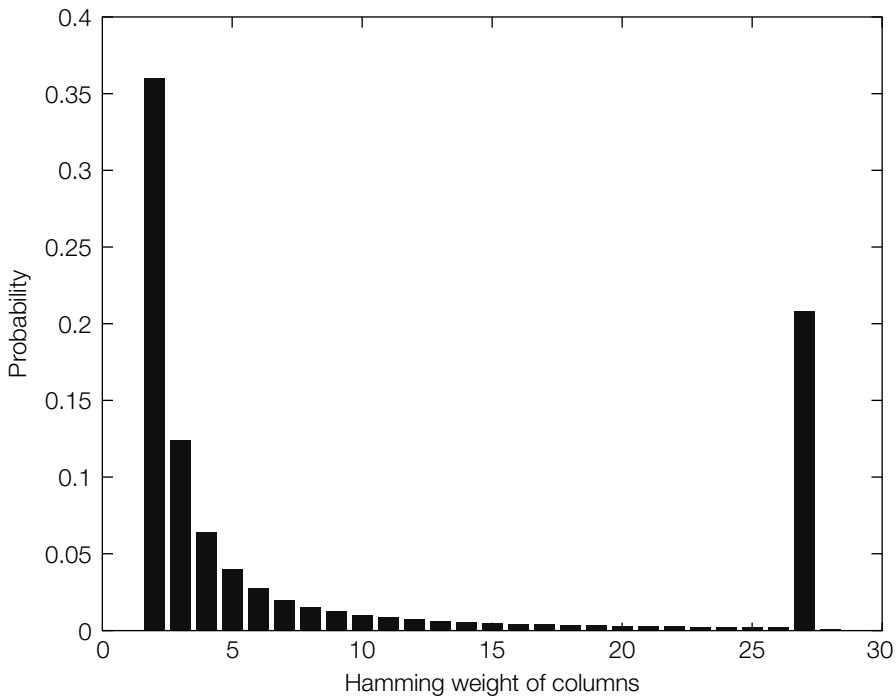

FIGURE 12.9

Robust soliton distribution for $k = 1{,}000$ and $c = 0.1$.

follows the distribution. Then, for each w_i, a column is generated by applying a random permutation to a column containing w_i ones and $k - w_i$ zeros.

It is known [275] that when the Hamming weights of the columns of **D** (and thus of **H**) follow the RSD distribution given by Equation 12.25, then the probability that the matrix LT process finishes successfully is greater than $1 - \delta$, provided that the message length, m, and the number of changeable pixels, k, satisfy the following inequality:

$$k > \theta m = m + O(\sqrt{m}\ln^2(m/\delta)). \tag{12.27}$$

The term $O(\sqrt{m}\ln^2(m/\delta))$ is the embedding capacity loss, since we are embedding m bits into $k > m$ changeable pixels. For $c = 0.1$, $\delta = 5$, and $m = 10{,}000$, the capacity loss is about 6% ($\theta = 1.062$) while the probability that the LT process succeeds is about 75%. This probability increases, and the capacity loss decreases, with increasing message length (see Table 12.2).

We now describe how this proposed coding can be incorporated into a steganographic method. The sender uses a secret stego key and forms the matrix **D** with columns following the RSD distribution. Then, he or she solves Equation 12.24 and finds the solution, **v**, using the Matrix LT procedure. The vector, **v**, tells the sender which pixels in the image should be modified.

The receiver forms the matrix **D** and extracts the message as $\mathbf{m} = \mathbf{Dy}$. However, in order to do so, the recipient needs to know the message length, m, because the RSD distribution, and thus **D**, depends on m. The remaining parameters c and δ can be publicly known.

We now discuss how to communicate the message length, m, to the receiver, together with other implementation details, such as what the sender should do if the matrix LT process fails. While there exist several different strategies, we describe the simplest one. The parameter, m, can be communicated simply by reserving a small portion of the cover image (e.g., determined from

Table 12.2 Running time (in seconds) for solving $m \times m$ and $m \times \theta m$ linear systems using Gaussian elimination and the matrix LT process, respectively ($c = 0.1$, $\delta = 5$); P is the probability of a successful pass through the LT process.

Runtime, overhead θ, and probability of successful pass				
m	**Gauss**	**LT**	θ	P
1,000	0.023	0.008	1.098	43%
10,000	17.4	0.177	1.062	75%
30,000	302	0.705	1.047	82%
100,000	9320	3.10	1.033	90%

the stego key) where we can use wet paper codes with a random matrix, **D**, that has a uniform distribution of 0s and 1s, and solve the system using Gaussian elimination. Since we only need, say, 20 bits to communicate m, solving a system of 20 equations using Gaussian elimination will be fast. The message itself is communicated in the remainder of the image using the matrix LT procedure whose matrix, **D**, follows the RSD distribution.

To deal with occasional failures of the matrix LT procedure, we can make **D** dependent on the message length, that is, make the seed for the pseudo-random number generator that is used to construct the RSD distribution be dependent on a combination of the stego key and message length. By doing so, we can simply add a dummy bit to the message, regenerate **D**, and repeat the embedding process until we obtain a successful pass through the matrix LT process.

The complexity of message embedding is $O(n\ln(k/\delta) + k\ln(k/\delta)) = O(n\ln(k/\delta))$ [150]. The first term arises from evaluating the product, **Dx**, while the second term is the complexity of the LT procedure. The gain in computational efficiency over regular Gaussian elimination, which has complexity $O(k^3)$, is substantial, as is demonstrated in the following Investigation and enumerated in Table 12.2.

INVESTIGATION

Matrix LT Process

In this investigation, we describe the LTSOLVER system that provides a fast solution to m linear equations, $\mathbf{Hv} = \mathbf{z}$, with k unknowns, $\theta m < k$, when the Hamming weights of the columns of **H** follow the RSD distribution of Equation 12.25.

System 20: SE_LTSOLVER

The function LTSOLVER accepts a binary $m \times k$ matrix **H** and a binary m-dimensional vector, **z**, as input and returns a solution, **v**, to the system using the Matrix LT procedure. It is based on the following algorithm:

0. $i = 0, t = 0$

1. WHILE $i < m$
{
$i = i + 1$

IF
there exists a column $i' \geq i$ with exactly one 1 in rows $j \geq i$.
THEN

swap the j-th and i-th rows and swap the corresponding
bits in the right hand side $\mathbf{z}[i]$ and $\mathbf{z}[j]$.
Also, swap the
i-th and i'-th columns and corresponding unknowns
$\mathbf{v}[i]$ and $\mathbf{v}[i']$.
Set $t = t + 1$ and store the transposition
$i \leftrightarrow i'$ as $\tau[t]$).

ELSE
declare a failure and STOP.

END WHILE

2. At this point, $\mathbf{H}$ has been permuted to the upper diagonal form.

Set $\mathbf{v}[i] = 0$ for $m < i \leq k$ and finish the solving
process using the usual back-substitution as in Gaussian elimination.

3. Apply the sequence of transpositions τ in the opposite order to $\mathbf{v}$,
$\mathbf{v} \leftarrow \tau[1](\tau[2](\ldots(\tau[t]\mathbf{v})\ldots))$.

4. The resulting $\mathbf{v}$ is the solution to $\mathbf{Hv} = \mathbf{z}$.

Experiment

The performance of the LTSOLVER was tested on matrices H of dimension $m \times \theta m$ whose column weights follow the RSD distribution with $c = 0.1$ and $\delta = 5$. Table 12.2 shows the running times for the Matrix LT process and the Gaussian elimination. It also shows the parameter, θ, and the probability of failure, P. The experiments were performed on a standard Pentium PC with a 3.4 MHz processor. The table clearly illustrates orders of magnitude reduction in computational time of the LT procedure over standard Gaussian elimination.

Perturbed Quantization

We now discuss perturbed quantization as a final example of steganographic algorithms based on nonshared selection rules. Interestingly, these techniques lead to some of the most secure steganographic schemes known today [148, 166].

We begin by generalizing the motivational example from Section 12.5.2 in which the sender embedded message bits during JPEG compression. The idea behind perturbed quantization is to couple the embedding procedure with common image-processing functions such as lossy compression, resampling, and filtering. This is accomplished by perturbing the final quantization step

associated with these operations. The sender's goal is to minimize the *combined* distortion due to embedding *and* image processing. This is an example of a steganographic algorithm with a nonshared selection rule, since the placement of embedding changes is determined by the original cover Work, which is a side information only available to the sender.

We first describe the general framework of perturbed quantization, followed by a heuristic security analysis. Let us assume that the cover Work is represented with a vector $\mathbf{x} \in \mathcal{P}^m$, where $\mathcal{P}$ is the range of its elements (pixels, coefficients, colors, indices). For example, for an 8-bit grayscale image, $\mathcal{P} = \{0, \ldots, 255\}$ and m is the number pixels. We further assume that F is an information-reducing process of the following form

$$F = Q \circ T : \mathcal{P}^m \rightarrow \mathcal{R}^n, \tag{12.28}$$

where $\mathcal{R}$ is the integer dynamic range of the processed signal, $\mathbf{y}$ (i.e., $\mathbf{y} = F(\mathbf{x})$). The output signal, $\mathbf{y}$, is represented by an n-dimensional vector of integers, $\mathbf{y} \in \mathcal{P}^n$, where $m > n$. The transform, $T : \mathcal{P}^m \rightarrow R^n$, is a real-valued transformation that is subsequently followed by a quantization, Q. The operator, $Q : R^n \rightarrow \mathcal{R}^n$, is the rounding to the closest integer within the range, $\mathcal{R}$. The intermediate image, $T(\mathbf{x})$, is denoted by an n-dimensional vector, $\mathbf{u} \in R^n$.

The sender identifies *changeable* elements, $\mathbf{u}[j], j \in \mathcal{J}$, whose values are close to the middle of the rounding intervals, as illustrated in Figure 12.10. Mathematically, we have

$$\mathcal{J} = \{j | j \in \{1, \ldots, n\}, \mathbf{u}[j] \in [L + 0.5 - \epsilon, L + 0.5 + \epsilon]\} \text{ for some integer } L. \tag{12.29}$$

The parameter ϵ is called the tolerance and controls which cover elements will be selected as changeable. This threshold can be adaptive and depend on the neighborhood of each cover element, or even depend on a secret key if desired. The sender communicates a message to the receiver by rounding the changeable elements, $\mathbf{u}[j], j \in \mathcal{J}$, to either L or $L + 1$ and rounding all other elements, $\mathbf{u}[i], i \in \mathcal{J}$, in a regular manner (i.e., $\mathbf{y}[i] = Q(\mathbf{u}[i])$).

Examples of possible operations, F, that are suitable for perturbed quantization include downsampling, resampling, and, of course, JPEG compression. In terms of the mathematical operations defined above, for downsampling, the

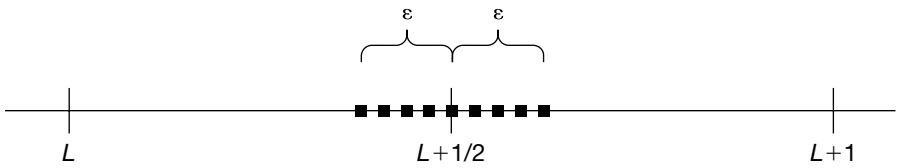

FIGURE 12.10

In perturbed quantization, values $\mathbf{u}[j]$ that fall in the dotted interval of length 2ϵ may be rounded to either L or $L + 1$.

transformation, T, maps a two-dimensional image array, $m_1 \times m_2$, of integers, $\mathbf{x}[i,j], i = 1, \ldots, m_1, j = 1, \ldots, m_2$, into an $n_1 \times n_2$ array of real numbers, $\mathbf{u} = \mathbf{u}[r,s], r = 1, \ldots, n_1, s = 1, \ldots, n_2, m_1 > n_1, m_2 > n_2$, using one of many resampling algorithms.

We now discuss the reasons why the nonshared selection rules used in perturbed quantization improve the security of steganographic schemes. To mount an attack, the warden must find statistical evidence that some of the values, $\mathbf{u}[j]$, have been quantized "incorrectly." However, for a number of reasons, this is difficult. First, the sender is using side information that is largely removed during quantization and is thus unavailable to the warden. Moreover, the rounding process at changeable elements is significantly influenced by the noise naturally present in images. This is less so for the remaining, unselected elements.

Nevertheless, the warden may be able to model some regions in the image sufficiently well (e.g., regions with a smooth gradient) that an attack on perturbed quantization becomes feasible. To avoid this, the sender can (and should) exploit additional selection rules to exclude from the changeable set, $\mathcal{J}$, those elements whose unquantized values can be predicted with high accuracy. There is no limit on the nature of these additional rules, since the sender and receiver can communicate using nonshared selection rules.

We note that the selection rule does not have to necessarily be of the type given in Equation 12.29, but can be defined differently based on heuristics depending on the format of the cover Work and properties of its elements. For example, in Fridrich *et al.* [148], the authors give an example of a slightly different selection rule for the situation when the information-reducing transformation is recompression of a cover JPEG image using a lower JPEG quality factor.

We now numerically evaluate the expected increase in distortion due to perturbed quantization steganography. Let us assume that the selection rule is of the form given by Equation 12.29. If the message bits are random, the result of embedding a message in the cover image, $\mathbf{x}$, is a probabilistic one-to-many mapping $[\mathbf{x} \rightarrow Q_\epsilon \circ T(\mathbf{x}) = \mathbf{y}']$, where $\mathbf{y}'$ is the stego image represented using an integer vector, $\mathbf{y}' \in \mathcal{R}^m$, and Q_ϵ is the perturbed quantizer, which is a probabilistic mapping

$$
\begin{aligned}
&Q_\epsilon(z) = L \quad \text{for } L \le z < L + 0.5 - \epsilon \\
&Q_\epsilon(z) = L + 1 \quad \text{for } L + 0.5 + \epsilon < z \le L + 1 \\
&Q_\epsilon(z) \in \{L, L+1\} \quad \text{with equal probability for } L + 0.5 - \epsilon \le z \le L + 0.5 + \epsilon.
\end{aligned}
$$

Note that $Q_\epsilon = Q$ for $\epsilon = 0$. The quantizers Q and Q_ϵ are identical with the exception of the interval $[L + 0.5 - \epsilon, L + 0.5 + \epsilon]$ where their outputs differ in 50% of the cases. It is straightforward to show that for a random variable, u, uniformly distributed on [0% , 1], the expected value of the quantization error, $|u - Q(u)|$, introduced by the regular rounding process, Q, is 1/4, while for the perturbed quantizer, Q_ϵ, it is $1/4 + \epsilon^2$. The average embedding distortion is the difference between the average error of both quantizers, which is ϵ^2. For

$\epsilon = 0.1$, this difference is at least one order of magnitude smaller than the average quantization error.

12.6 SUMMARY

This chapter has focused on a specific data-hiding application—steganography. The main points of this chapter are the following.

- As opposed to digital watermarking, the main property of steganography is statistical undetectability of embedded data. The payload is usually unrelated to the cover Work, which only serves as a decoy.
- The information-theoretic definition of steganographic security (Cachin's definition) is the most widely used definition in practice. It is usually applied in a simplified form by accepting a model for the cover Work.
- The simplest steganographic method—the LSB embedding—embeds messages by flipping the LSBs of individual elements of the cover Work. LSB embedding is not secure with respect to Cachin's definition because it introduces characteristic artifacts into the histogram of pixel values.
- Secure steganographic schemes must take into account steganalytic methods. One possibility is to replace the embedding operation of LSB flipping (F5) to avoid introducing easily detectable artifacts. Another possibility is to mask the embedding distortion as a naturally occurring phenomenon, such as to occur during image acquisition (stochastic modulation). Alternatively, we can design schemes that preserve some vital statistical characteristics of the cover image (OutGuess) or a *model* of the cover that is recoverable from the stego Work (model-based steganography).
- The embedding efficiency of steganographic schemes is defined as the average number of random message bits embedded using one embedding change. If the secret message is shorter than the embedding capacity, the embedding efficiency can be substantially improved using matrix embedding. This general embedding principle can be applied to the majority of steganographic schemes.
- In a typical steganographic scheme, the placement of embedding changes (the selection rule) is shared between the sender and the recipient. However, there are many situations when this information cannot be shared, such as in adaptive steganography, selection rules determined from side-information, or in public-key steganography. The problem of nonshared selection rules is equivalent to writing in memory with defective cells and can be efficiently approached using sparse linear codes, known as LT codes.