Chapter 3

A survey of steganographic techniques

Neil F. Johnson and Stefan C. Katzenbeisser

Many different steganographic methods have been proposed during the last few years; most of them can be seen as substitution systems. Such methods try to substitute redundant parts of a signal with a secret message (as presented in Section 2.3); their main disadvantage is the relative weakness against cover modifications. Recently, the development of new robust watermarking techniques led to advances in the construction of robust and secure steganography systems. Therefore, some of the methods presented here are strongly related to watermarking techniques of Chapter 6.

There are several approaches in classifying steganographic systems. One could categorize them according to the type of covers used for secret communication. A classification according to the cover modifications applied in the embedding process is another possibility. We want to follow the second approach and group steganographic methods in six categories, although in some cases an exact classification is not possible:

- **Substitution systems** substitute redundant parts of a cover with a secret message;
- **Transform domain techniques** embed secret information in a transform space of the signal (e.g., in the frequency domain);
- **Spread spectrum techniques** adopt ideas from spread spectrum communication;

- **Statistical methods** encode information by changing several statistical properties of a cover and use hypothesis testing in the extraction process;
- **Distortion techniques** store information by signal distortion and measure the deviation from the original cover in the decoding step;
- **Cover generation methods** encode information in the way a cover for secret communication is created.

In the following sections these six categories will be discussed.

3.1 PRELIMINARY DEFINITIONS

Throughout the following sections we want to refer to the cover used in the embedding step as c. We will further assume (without loss of generality) that any cover can be represented by a sequence of numbers c_i of length $\ell(c)$ (i.e., $1 \leq i \leq \ell(c)$). In the case of digital sound this could be just the sequence of samples over time; in the case of a digital image, a sequence can be obtained by vectorizing the image (i.e., by lining up all pixels in a left-to-right and top-to-bottom order). Possible values of c_i are $\{0, 1\}$ in the case of binary images or integers greater than 0 and less than 256 in the case of quantized images or sound. We will denote the stego-object by s which is again a sequence s_i of length $\ell(c)$.

Sometimes we have to index all cover-elements c_i; we will use the symbol j for such an index. If the index is itself indexed by some set, we use the notation j_i. When we refer to the j_ith cover-element we mean c_{j_i}. We will refer to a stego-key as k; the structure of k will be explained separately in each steganographic application. The secret message will be denoted by m, the length of m by $\ell(m)$, and the bits forming m by m_i, $1 \leq i \leq \ell(m)$. Unless otherwise stated, we assume that $m_i \in \{0, 1\}$.

A color value is normally a three-component vector in a *color space* (a set of possible colors), see [1]. A well-known color space is RGB. Since the colors red, green, and blue are *additive primaries*, every color can be specified as a weighted sum of a red, green, and a blue component. A vector in RGB space describes the intensities of these components. Another space, known as YCbCr, distinguishes between a luminance (Y) and two chrominance (Cb,Cr) components. Whereas the Y component accounts for the brightness of a color, Cb and Cr distinguish between the color grades. A color vector in RGB can be converted to YCbCr using the transform:

$$\begin{aligned} Y &= 0.299\,R + 0.587\,G + 0.114\,B \\ Cb &= 0.5 + (B - Y)/2 \\ Cr &= 0.5 + (R - Y)/1.6 \end{aligned} \tag{3.1}$$

An *image* C is a discrete function assigning a color vector (of any color space) $c(x, y)$ to every pixel (x, y).

3.2 SUBSTITUTION SYSTEMS AND BITPLANE TOOLS

A number of methods exist for hiding information in various media. These methods range from LSB coding—also known as bitplane or noise insertion tools—manipulation of image or compression algorithms to modification of image properties such as luminance. Basic substitution systems try to encode secret information by substituting insignificant parts of the cover by secret message bits; the receiver can extract the information if he has knowledge of the positions where secret information has been embedded. Since only minor modifications are made in the embedding process, the sender assumes that they will not be noticed by a passive attacker.

3.2.1 Least significant bit substitution

Bitplane tools encompass methods that apply LSB insertion and noise manipulation. These approaches are common in steganography and are relatively easy to apply in image and audio [2–6]. A surprising amount of information can be hidden with little, if any, perceptible impact to the carriers [5, 7, 8].

Sample tools used in this group include StegoDos [9], S-Tools [10], Mandelsteg [11], EzStego [12], Hide and Seek [13], Hide4PGP [14], White Noise Storm [15], and Steganos [16]. The image formats typically used in such steganography methods are lossless and the data can be directly manipulated and recovered. Some of these programs apply compression and encryption in addition to steganography services. These services provide better security of the hidden data. Even so, the bitplane methods are rather brittle and vulnerable to corruption due to small changes to the carrier.

The embedding process consists of choosing a subset $\{j_1, \ldots, j_{\ell(m)}\}$ of cover-elements and performing the substitution operation $c_{j_i} \leftrightharpoons m_i$ on them, which exchanges the LSB of c_{j_i} by m_i (m_i can either be 1 or 0). One could also imagine a substitution operation which changes more than one bit of the cover, for instance by storing two message bits in the two least significant bits of one cover-element. In the extraction process, the LSB of the selected cover-elements are extracted and lined up to reconstruct the secret message. This basic scheme is presented in Algorithms 3.1 and 3.2. One problem remains to be solved: in which way should the c_{j_i} be chosen?

In order to be able to decode the secret message, the receiver must have access to the sequence of element indices used in the embedding process. In the simplest

Algorithm 3.1 Embedding process: least significant bit substitution

for $i = 1, \ldots, \ell(c)$ **do**
  $s_i \leftarrow c_i$
end for
for $i = 1 \ldots, \ell(m)$ **do**
  compute index j_i where to store ith message bit
  $s_{j_i} \leftarrow c_{j_i} \leftrightharpoons m_i$
end for

Algorithm 3.2 Extraction process: least significant bit substitution

for $i = 1, \ldots, \ell(M)$ **do**
  compute index j_i where the ith message bit is stored
  $m_i \leftarrow \mathrm{LSB}(c_{i_j})$
end for

case, the sender uses all cover-elements for information transfer, starting at the first element. Since the secret message will normally have less bits than $\ell(c)$, the embedding process will be finished long before the end of the cover. In this case, the sender can leave all other cover elements unchanged. This can, however, lead to a serious security problem: the first part of the cover will have different statistical properties than the second part, where no modifications have been made. To overcome this problem, for instance the public domain program PGMStealth enlarges the secret message with random bits—so that $\ell(c) = \ell(m)$—in an attempt to create an equal change in randomness at the beginning and the end of the cover. The embedding process thus changes far more elements than the transmission of the secret would require. Therefore the probability that an attacker will suspect secret communication increases.

A more sophisticated approach is the use of a pseudorandom number generator to spread the secret message over the cover in a rather random manner; a popular approach is the *random interval method* (e.g., [3]). If both communication partners share a stego-key k usable as a seed for a random number generator, they can create a random sequence $k_1, \ldots, k_{\ell(m)}$ and use the elements with indices

$$\begin{aligned} j_1 &= k_1 \\ j_i &= j_{i-1} + k_i, \quad i \geq 2 \end{aligned} \tag{3.2}$$

for information transfer. Thus, the distance between two embedded bits is determined pseudorandomly. Since the receiver has access to the seed k and knowledge of the pseudorandom number generator, he can reconstruct k_i and therefore the entire sequence of element indices j_i. This technique—which is especially efficient

Algorithm 3.3 Embedding process: random interval method

for $i = 1 \ldots, \ell(c)$ **do**
  $s_i \leftarrow c_i$
end for
generate random sequence k_i using seed k
$n \leftarrow k_1$
for $i = 1, \ldots, \ell(m)$ **do**
  $s_n \leftarrow c_n \leftrightharpoons m_i$
  $n \leftarrow n + k_i$
end for

Algorithm 3.4 Extraction process: random interval method

generate random sequence k_i using seed k
$n \leftarrow k_1$
for $i = 1, \ldots, \ell(m)$ **do**
  $m_i \leftarrow \mathrm{LSB}(c_n)$
  $n \leftarrow n + k_i$
end for

in the case of stream covers—is illustrated in Algorithms 3.3 and 3.4, which are special cases of the general framework presented in Algorithms 3.1 and 3.2.

3.2.2 Pseudorandom permutations

If all cover bits can be accessed in the embedding process (i.e., if c is a random access cover), the secret message bits can be distributed randomly over the whole cover. This technique further increases the complexity for an attacker, since it is not guaranteed that subsequent message bits are embedded in the same order.

In a first attempt Alice could create (using a pseudorandom number generator) a sequence $j_1, \ldots, j_{\ell(m)}$ of element indices and store the kth message bit in the element with index j_k. Note that one index could appear more than once in the sequence, since we have not restricted the output of the pseudorandom number generator in any way. We call such a case "collision." If a collision occurs, Alice will possibly try to insert more than one message bit into one cover-element, thereby corrupting some of them. If the message is quite short compared with the number of cover-elements, she hopes that the probability of collisions is negligible and that corrupted bits could be reconstructed using an error-correcting code. This is, how-

ever, only the case for quite short secret messages. The probability p of at least one collision can be estimated[1] by (provided that $\ell(m) \ll \ell(c)$):

$$p \approx 1 - \exp\left(-\frac{\ell(m)[\ell(m)-1]}{2\ell(c)}\right)$$

For constant $\ell(c)$, p converges rapidly to 1 as $\ell(m)$ increases. If, for example, a digital image with 600×600 pixels is used as cover and about 200 pixels are selected in the embedding process, p is approximately 5%. On the other hand, if 600 pixels are used for information transfer, p increases to about 40%. We can conclude that only for very short messages the probability of collisions is negligible; if the message size increases, collisions must definitely be taken into account.

To overcome the problem of collisions, Alice could keep track of all cover-bits which have already been used for communication in a set B. If during the embedding process one specific cover-element has not been used prior, she adds its index to B and continues to use it. If, however, the index of the cover-element is already contained in B, she discards the element and chooses another cover-element pseudorandomly. At the receiver side, Bob applies a similar technique.

Another method has been proposed by Aura [18]; he uses the basic substitution scheme of Algorithms 3.1 and 3.2 and calculates the index j_i via a pseudorandom permutation of the set $\{1, \ldots, \ell(c)\}$. Suppose the number $\ell(c)$ can be expressed as a product of two numbers, X and Y (recall that this is always the case for digital images), and h_K is an arbitrary cryptographically secure hash function depending on a key k. Let k_1, k_2 and k_3 be three secret keys. It can then be shown [19, 20] that Algorithm 3.5 outputs a different number j_i for each input i ($1 \leq i \leq XY$), (i.e., it produces a pseudorandom permutation of the set $\{1, \ldots, \ell(c)\}$), provided that the algorithm is evaluated with input $i = 1, \ldots, \ell(c)$.

Alice first splits the stego-key k into three pieces k_1, k_2 and k_3. In the embedding process she stores the ith message bit in the element with index j_i, which is computed according to Algorithm 3.5. Collisions do not occur, since Algorithm 3.5 does not produce duplicate element indices. If Bob has access to the three keys k_1, k_2 and k_3, he is able to reconstruct the positions where Alice embedded the secret

1 The problem of calculating p is an instance of the so-called birthday paradox: an urn is filled with n balls, numbered from 1 to n. Suppose that m balls are drawn from the urn with replacement and their numbers are listed. The probability $P(n, m)$ that at least one ball is drawn twice, provided that $m = O(\sqrt{n})$, is given by [17]

$$P(n,m) = 1 - \prod_{i=0}^{m-1}\left(1 - \frac{i}{n}\right) \to 1 - \exp\left(-\frac{m(m-1)}{2n} + O\left(\frac{1}{\sqrt{n}}\right)\right)$$

Algorithm 3.5 Computing the index j_i using pseudorandom permutations

$v \leftarrow i \operatorname{div} X$
$u \leftarrow i \bmod X$
$v \leftarrow (v + h_{k_1}(u)) \bmod Y$
$u \leftarrow (u + h_{k_2}(v)) \bmod X$
$v \leftarrow (v + h_{k_3}(u)) \bmod Y$
$j_i \leftarrow vX + u$

message bits. However, Aura's method needs a considerable amount of computation time, since the chosen hash function must be evaluated $3\ell(m)$ times.

3.2.3 Image downgrading and covert channels

In 1992, Kurak and McHugh [5] reported on a security threat in high-security operating systems. Their fear was that a steganographic technique, called image downgrading, could be used to exchange images covertly. Image downgrading is a special case of a substitution system in which images act both as secret messages and covers. Given a cover-image and a secret image of equal dimensions, the sender exchanges the four least significant bits of the cover's grayscale (or color) values with the four most significant bits of the secret image. The receiver extracts the four least significant bits out of the stego-image, thereby gaining access to the most significant bits of the secret image. While the degradation of the cover is not visually noticeable in many cases, 4 bits are sufficient to transmit a rough approximation of the secret image.

In multilevel-secure operating systems, subjects (processes, users) and objects (files, databases, etc.) are assigned a specific security level; for example, see the famous Bell-LaPadula [21] model. Subjects are normally only allowed to access objects with a lower security level ("no read up"), whereas they are only able to write onto objects with a higher level ("no write down"). Whereas the reason for the first restriction is obvious, the second attempts to prohibit users from making confident information available to subjects with a lower security classification. Information downgrading can be used to declassify or *downgrade* information (hence the name) by embedding classified information into objects with a substantially lower security classification and thus subvert the principle of "no write down." Chapter 4 will look at possible counterstrategies.

3.2.4 Cover-regions and parity bits

We will call any nonempty subset of $\{c_1, \ldots, c_{\ell(c)}\}$ a cover-region. By dividing the cover in several disjoint regions, it is possible to store one bit of information in a whole cover-region rather than in a single element. A *parity bit* of a region I can be calculated by

$$p(I) = \sum_{j \in I} LSB(c_j) \bmod 2 \tag{3.3}$$

In the embedding step, $\ell(m)$ disjoint cover-regions I_i $(1 \leq i \leq \ell(m))$ are selected, each encodes one secret bit m_i in the parity bit $p(I_i)$. If the parity bit of one cover-region I_i does not match with the secret bit m_i to encode, one LSB of the values in I_i is flipped. This will result in $p(I_i) = m_i$. In the decoding process, the parity bits of all selected regions are calculated and lined up to reconstruct the message. Again, the cover-regions can be constructed pseudorandomly using the stego-key as a seed.

Although the method is not more robust than simple bit substitution, it is conjectured to be more powerful in many cases. First, the sender can choose which element should be modified in the cover-region; he can do it in a way that changes the cover statistics least. Furthermore, the probability p_0^* that the parity bit of a cover-region consisting of N randomly chosen elements is zero, is approximately $1/2$, nearly independent of the probability p_0 that the LSB of one randomly selected cover-element is zero, since

$$\begin{aligned} p_0^* &= \sum_{i=0}^{\lfloor N/2 \rfloor} \binom{N}{2i} (1-p_0)^{2i} p_0^{N-2i} \\ &= \frac{p_0^N}{2} \left[\left(1 + \frac{1-p_0}{p_0} \right)^N + \left(1 - \frac{1-p_0}{p_0} \right)^N \right] \\ &= \frac{1}{2} \left(1 + (2p_0 - 1)^N \right) \end{aligned} \tag{3.4}$$

Equation (3.4) follows from the fact that $p(I) = 0$ if and only if there is an even number of pixels in the cover-region which have least significant bit 1. Since $(2p_0 - 1)^N \to 0$ if $0 < p_0 < 1$, we can conclude that p_0^* rapidly approaches $1/2$ as N increases, regardless of p_0. This indicates that the effect of the embedding process on the cover can be reduced by increasing N.

3.2.5 Palette-based images

In a palette-based image only a subset of colors from a specific color space can be used to colorize the image. Every palette-based image format consists of two parts: a *palette* specifying N colors as a list of indexed pairs $(i, \mathbf{c}_i)$, assigning a color vector $\mathbf{c}_i$ to every index i, and the actual image data which assign a palette index to every pixel rather than the color value itself. If only a small number of color values are used throughout the image, this approach greatly reduces the file size. Two of the most popular formats are the graphics interchange format (GIF) and the BMP bitmap format. However, due to the availability of sophisticated compression techniques, their use declines.

Generally, there are two ways to encode information in a palette-based image: either the palette or the image data can be manipulated. The LSB of the color vectors could be used for information transfer, just like the substitution methods presented in the last subsections. Alternatively, since the palette does not need to be sorted in any way, information can be encoded in the way the colors are stored in the palette. Since there are $N!$ different ways to sort the palette, there is enough capacity to encode a small message. However, all methods which use the order of a palette to store information, are not robust, since an attacker can simply sort the entries in a different way and destroy the secret message (he thereby does not even modify the picture visibly).

Alternatively, information can be encoded in the image data. Since neighboring palette color values need not be perceptually similar, the approach of simply changing the LSB of some image data fails. Some steganographic applications (e.g., the program EzStego) therefore sort the palette so that neighboring colors are perceptually similar before they start the embedding process. Color values can, for instance, be stored according to their Euclidian distance in RGB space:

$$d = \sqrt{R^2 + G^2 + B^2} \tag{3.5}$$

Since the human visual system is more sensitive to changes in the luminance of a color, another (probably better) approach would be sorting the palette entries according to their luminance component, see (3.1). After the palette is sorted, the LSB of color indices can safely be altered.

Fridrich [22] proposes using a slightly different technique which does not need the palette to be sorted: for every pixel, the set of closest colors (in the Euclidian norm) is calculated. Starting with the closest color, the sender proceeds to find the next-closest color until a color is found where its parity $(R + G + B \bmod 2)$ matches with the secret bit to encode. Once such a color is found, the pixel is changed to this new color.

Yet another steganographic application reduces the total number of color values

in a picture to $\lfloor N/2 \rfloor$ using some dithering method, and doubles the entire palette; thereby all doubled entries are slightly modified. After this preprocessing stage, each color value of the dithered image corresponds to two palette entries, from which one is chosen according to a secret message bit (e.g., Mandelsteg [11], S-Tools [10], Hide4PGP [14], and Hide and Seek [13] apply variations of this method).

3.2.6 Quantization and dithering

Dithering and quantization of digital images can be used for embedding secret information. Matsui and Tanaka [23] presented two steganographic systems which operate on quantized images. We briefly review quantization in the context of predictive coding here. In predictive coding, the intensity of each pixel is predicted based on the pixel values in a specific neighborhood; the prediction may be a linear or nonlinear function of the surrounding pixel values. In its simplest form, the difference e_i between adjacent pixels x_i and x_{i+1} is calculated and fed into a quantizer Q which outputs a discrete approximation Δ_i of the difference signal $x_i - x_{i-1}$ (i.e., $\Delta_i = Q(x_i - x_{i-1})$). Thus, in each quantization step a quantization error is introduced. For highly correlated signals we can expect Δ_i to be close to zero, so an entropy coder—which tries to create a minimum-redundancy code given a stochastic model of the data to be transmitted—will be efficient. At the receiver side the difference signal is dequantized and added to the last signal sample in order to construct an estimate for the sequence x_i.

For steganographic purposes the quantization error in a predictive coding scheme can be utilized; specifically, we adjust the difference signal Δ_i so that it transmits additional information. In this scheme, the stego-key consists of a table which assigns a specific bit to every possible value of Δ_i; for instance, the following assignment could be made:

Δ_i	-4	-3	-2	-1	0	1	2	3	4
	0	1	0	1	1	1	0	0	1

In order to store the ith message bit in the cover-signal, the quantized difference signal Δ_i is computed. If Δ_i does not match (according to the secret table) with the secret bit to be encoded, Δ_i is replaced by the nearest Δ_j where the associated bit equals the secret message bit. The resulting values Δ_i are then fed into the entropy coder. At the receiver side, the message is decoded according to the difference signal Δ_i and the stego-key.

Secret information can also be inserted into a signal during a dithering process; see [23] and Baharav and Shaked [24] for details.

Algorithm 3.6 Zhao and Koch's algorithm for data embedding in binary images

for $i = 1, \ldots, \ell(M)$ **do**
  do forever
    pseudorandomly select a new image block B_j
    / Test, if block B_i is valid */*
    if $P_1(B_j) > R_1 + 3\lambda$ or $P_1(B_j) < R_0 - 3\lambda$ **then continue**
    if ($c_i = 1$ and $P_1(B_j) < R_0$) or ($c_i = 0$ and $P_1(B_j) > R_1$) **then**
      mark block B_j as unusable, i.e. modify block so that
        either $P_1(B_j) < R_0 - 3\lambda$ or $P_1(B_j) > R_1 + 3\lambda$
      continue
    endif
    break
  enddo
  / Embed secret message bit in B_j */*
  if $c_i = 1$ **then**
    modify B_j so that $P_1(B_j) \geq R_1$ and $P_1(B_j) \leq R_1 + \lambda$
  else
    modify B_j so that $P_0(B_j) \leq R_0$ and $P_0(B_j) \geq R_0 - \lambda$
  end if
end for

3.2.7 Information hiding in binary images

Binary images—like digitized fax data—contain redundancies in the way black and white pixels are distributed. Although the implementation of a simple substitution scheme is possible (e.g., certain pixels could be set to black or white depending on a specific message bit), these systems are highly susceptible to transmission errors and are therefore not robust.

One information hiding scheme which uses the number of black pixels in a specific image region to encode secret information was presented by Zhao and Koch [25]. A binary image is divided into rectangular image blocks B_i; let $P_0(B_i)$ be the percentage of black pixels in the image block B_i and $P_1(B_i)$ the percentage of white pixels, respectively. Basically, one block embeds a 1, if $P_1(B_i) > 50\%$ and a 0, if $P_0(B_i) > 50\%$. In the embedding process the color of some pixels is changed so that the desired relation holds. Modifications are carried out at those pixels whose neighbors have the opposite color; in sharply contrasted binary images, modifications are carried out at the boundaries of black and white pixels. These rules assure that the modifications are not generally noticeable.

In order to make the entire system robust to transmission errors and other

Algorithm 3.7 Extraction process (Zhao and Koch)

```
for i = 1, ..., ℓ(M) do
  do forever
    pseudorandomly select image block B_j
    if P_1(B_j) > R_1 + 3λ or P_1(B_j) < R_0 − 3λ then continue
    break
  enddo
  if P_1(B_j) > 50% then
    m_i ← 1
  else
    m_i ← 0
  end if
end for
```

image modifications, we have to adapt the embedding process. If it is possible that some pixels change color during the transmission process, it could be the case that for instance $P_1(B_i)$ drops from 50.6% to 49.5%, thereby destroying the embedded information. Therefore two threshold values $R_1 > 50\%$ and $R_0 < 50\%$ and a robustness parameter λ, which specifies the percentage of pixels which can change color during transmission, are introduced. The sender assures during the embedding process that either $P_1(B_i) \in [R_1, R_1 + \lambda]$ or $P_0(B_i) \in [R_0 - \lambda, R_0]$ instead of $P_1(B_i) > 50\%$ and $P_0(B_i) < 50\%$. If too many pixels must be changed in order to achieve that goal, the block is marked as "invalid": $P_1(B_i)$ is modified to fulfill one of the two conditions

$$\begin{aligned} P_1(B_i) &< R_0(B_i) - 3\lambda \\ P_1(B_i) &> R_1(B_i) + 3\lambda \end{aligned}$$

and another block is pseudorandomly chosen for bit i. In the decoding process, invalid blocks are skipped. Otherwise, the information is decoded according to $P_1(B_i)$. The embedding and extraction algorithms are outlined in Algorithms 3.6 and 3.7.

A different embedding scheme, presented by Matsui and Tanaka [23], uses the lossless compression system which is used to encode information in a facsimile document. According to a recommendation of the former Comité Consultatif International Télégraphique et Téléphonique (which is now the International Telecommunication Union) [26], fax images can be coded using a combination of run length (RL) and Huffman encoding. RL techniques utilize the fact that in a binary image successive pixels have the same color with high probability. Figure 3.1 shows one scan line from a fax document; we will indicate positions with changing colors with

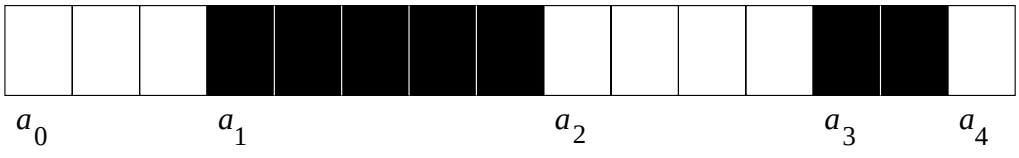

Figure 3.1 One scan line of a binary image.

a_i. Instead of coding the color of every pixel explicitly, RL methods code the positions of color changes (a_i) together with the number $RL(a_i, a_{i+1})$ of successive pixels with the same color starting at a_i. Our hypothetical scan line of Figure 3.1 would be coded by $\langle a_0, 3\rangle, \langle a_1, 5\rangle, \langle a_2, 4\rangle, \langle a_3, 2\rangle, \langle a_4, 1\rangle$. We can thus describe a binary image as a sequence of RL elements $\langle a_i, RL(a_i, a_{i+1})\rangle$.

Information can be embedded into a binary, run-length encoded image by modifying the least significant bit of $RL(a_i, a_{i+1})$. In the encoding process we modify the run lengths of the binary picture so that $RL(a_i, a_{i+1})$ is even, if the ith secret message bit m_i is zero. If, however, $RL(a_i, a_{i+1})$ is odd, m_i is one. This can be achieved, for example, by the following manner: if m_i is zero but $RL(a_i, a_{i+1})$ is odd, we move the position of a_{i+1} one pixel to the left. On the other hand, we move a_{i+1} one pixel to the right, if $m_i = 1$ and $RL(a_i, a_{i+1})$ is even. This insertion technique, however, leads to problems if the run-length $RL(a_i, a_{i+1})$ is one. If the run-length needs to be changed in the embedding process, it could be lost. We therefore have to assure that such a situation will never happen; for example, all RL elements with run-length one could be dropped before starting the embedding process.

3.2.8 Unused or reserved space in computer systems

Taking advantage of unused or reserved space to hold covert information provides a means of hiding information without perceptually degrading the carrier. For example: the way operating systems store files typically results in unused space that appears to be allocated to a file. For example, under Windows 95 operating system, drives formatted as FAT16 (MS-DOS compatible) without compression typically use cluster sizes of 32^2 kilobytes (Kb). This means that the minimum space allocated to a file is 32 Kb. If a file is 1 Kb in size, then an additional 31 Kb is "wasted." This "extra" space can be used to hide information without showing up in the directory. Unused space in file headers of image and audio can also be used to hold "extra" information.

2 This depends on the size of the hard drive.

Another method of hiding information in file systems is to create a hidden partition. These partitions are not seen if the system is started normally. However, in many cases, running a disk configuration utility (such as DOS's FDISK) exposes the hidden partition. These concepts have been expanded in a novel proposal of a steganographic file system [27, 28]. If the user knows the file name and password, access is granted to the file; otherwise, no evidence of the file exists in the system.

Protocols in the OSI network model have characteristics that can be used to hide information [29]. TCP/IP packets used to transport information across the Internet have unused space in the packet headers. The TCP packet header has six unused (reserved) bits and the IP packet header has two reserved bits. Thousands of packets are transmitted with each communication channel, which provides an excellent covert communication channel if unchecked. The ease in use and abundant availability of steganography tools has law enforcement concerned in trafficking of illicit material via Web page images, audio, and other files being transmitted through the Internet. Methods of message detection and understanding the thresholds of current technology are necessary to uncover such activities (see Chapter 4).

3.3 TRANSFORM DOMAIN TECHNIQUES

We have seen that LSB modification techniques are easy ways to embed information, but they are highly vulnerable to even small cover modifications. An attacker can simply apply signal processing techniques in order to destroy the secret information entirely. In many cases even the small changes resulting out of lossy compression systems yield to total information loss.

It has been noted early in the development of steganographic systems that embedding information in the frequency domain of a signal can be much more robust than embedding rules operating in the time domain. Most robust steganographic systems known today actually operate in some sort of transform domain.

Transform domain methods hide messages in significant areas of the cover image which makes them more robust to attacks, such as compression, cropping, and some image processing, than the LSB approach. However, while they are more robust to various kinds of signal processing, they remain imperceptible to the human sensory system. Many transform domain variations exist. One method is to use the discrete cosine transformation (DCT) [30–33] as a vehicle to embed information in images; another would be the use of wavelet transforms [34]. Transformations can be applied over the entire image [30], to blocks throughout the image [35, 36], or other variations. However, a trade-off exists between the amount of information added to the image and the robustness obtained [7, 37]. Many transform domain methods

are independent to image format and may survive conversion between lossless and lossy formats.

Before we describe transform domain steganographic methods, we will briefly review the Fourier and cosine transforms which can be used to map a signal into the frequency domain. The discrete Fourier transform (DFT) of a sequence s of length N is defined to be

$$S(k) = \mathcal{F}\{s\} = \sum_{n=0}^{N-1} s(n) \exp\left(-\frac{2in\pi k}{N}\right) \tag{3.6}$$

where $i = \sqrt{-1}$ is the imaginary unit. The inverse Fourier transform is given by

$$s(k) = \mathcal{F}^{-1}\{S\} = \sum_{n=0}^{N-1} S(n) \exp\left(\frac{2in\pi k}{N}\right) \tag{3.7}$$

Another useful transform is the DCT, given by

$$\begin{aligned} S(k) &= \mathcal{D}\{s\} = \frac{C(k)}{2} \sum_{j=0}^{N} s(j) \cos\left(\frac{(2j+1)k\pi}{2N}\right) \\ s(k) &= \mathcal{D}^{-1}\{S\} = \sum_{j=0}^{N} \frac{C(j)}{2} s(j) \cos\left(\frac{(2j+1)k\pi}{2N}\right) \end{aligned} \tag{3.8}$$

where $C(u) = 1/\sqrt{2}$ if $u = 0$ and $C(u) = 1$ otherwise. The DCT has the primary advantage that $\mathcal{D}\{s\}$ is a sequence of real numbers, provided that the sequence s is real. In digital image processing, the two-dimensional version of the DCT is used:

$$\begin{aligned} S(u,v) &= \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} s(x,y) \cos\left(\frac{\pi u(2x+1)}{2N}\right) \cos\left(\frac{\pi v(2y+1)}{2N}\right) \\ s(x,y) &= \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)S(u,v) \cos\left(\frac{\pi u(2x+1)}{2N}\right) \cos\left(\frac{\pi v(2y+1)}{2N}\right) \end{aligned}$$

The two-dimensional DCT is the "heart" of the most popular lossy digital image compression system used today: the JPEG system [38, 39] (see Figure 3.2). JPEG first converts the image to be compressed into the YCbCr color space and breaks up each color plane into 8×8 blocks of pixels. Then, all blocks are DCT transformed. In a quantization step all DCT coefficients are divided by some predefined quantization values (see Table 3.1) and rounded to the nearest integer (according to a *quality factor*, the quantization values can be scaled by a constant). The purpose of this process is to modulate the influence of the different spectral components on the

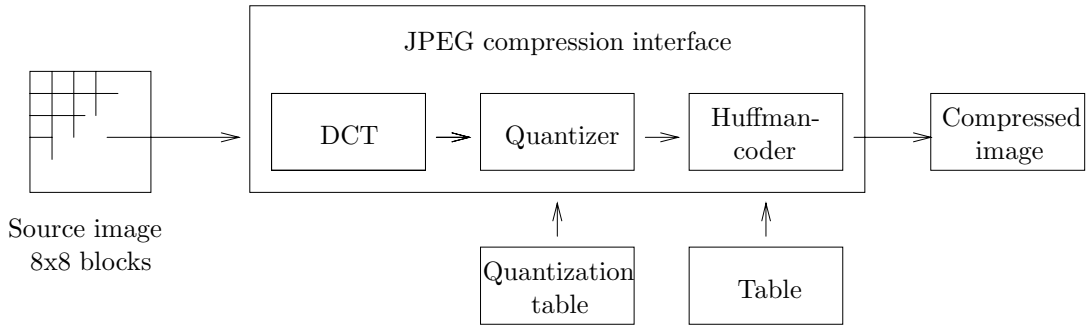

Figure 3.2 Outline of the JPEG image compression algorithm.

$(\mathbf{u}, \mathbf{v})$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	16	11	10	16	24	40	51	61
1	12	12	14	19	26	58	60	55
2	14	13	16	24	40	57	69	56
3	14	17	22	29	51	87	80	62
4	18	22	37	56	68	109	103	77
5	24	35	55	64	81	104	113	92
6	49	64	78	87	103	121	120	101
7	72	92	95	98	112	100	103	99

Table 3.1 Quantization values used in the JPEG compression scheme (luminance components).

image. In particular, the influence of the highest DCT coefficients is reduced: they are likely to be dominated by noise and are not expected to contribute significant details to the picture. The resulting quantized DCT coefficients are compressed using an entropy coder (e.g., Huffman [40] or arithmetic coding). In the JPEG decoding step all DCT coefficients are dequantized (i.e., multiplied with the quantization values which had been used in the encoding step). Afterwards an inverse DCT is performed to reconstruct the data. The restored picture will be close to (but not identical with) the original one; but if the quantization values were set properly, there should be no noticeable difference for a human observer.

3.3.1 Steganography in the DCT domain

One popular method of encoding secret information in the frequency domain is modulating the relative size of two (or more) DCT coefficients within one image

Algorithm 3.8 DCT–Steg encoding process

for $i = 1, \ldots, \ell(M)$ **do**
  choose one cover-block b_i
  $B_i = \mathcal{D}\{b_i\}$
  if $m_i = 0$ **then**
    if $B_i(u_1, v_1) > B_i(u_2, v_2)$ **then**
      swap $B_i(u_1, v_1)$ and $B_i(u_2, v_2)$
    end if
  else
    if $B_i(u_1, v_1) < B_i(u_2, v_2)$ **then**
      swap $B_i(u_1, v_1)$ and $B_i(u_2, v_2)$
    end if
  end if
  adjust both values so that $|B_i(u_1, v_1) - B_i(u_2, v_2)| > x$
  $b'_i = \mathcal{D}^{-1}\{B_i\}$
end for
create stego-image out of all b'_i

block. We will describe a system which uses digital images as covers and which is similar to a technique proposed by Zhao and Koch [25].

During the encoding process, the sender splits the cover-image in 8×8 pixel blocks; each block encodes exactly one secret message bit. The embedding process starts with selecting a pseudorandom block b_i which will be used to code the ith message bit. Let $B_i = \mathcal{D}\{b_i\}$ be the DCT-transformed image block.

Before the communication starts, both sender and receiver have to agree on the location of two DCT coefficients, which will be used in the embedding process; let us denote these two indices by (u_1, v_1) and (u_2, v_2). The two coefficients should correspond to cosine functions with middle frequencies; this ensures that the information is stored in significant parts of the signal (hence the embedded information will not be completely damaged by JPEG compression). Furthermore, we can assume that the embedding process will not degenerate the cover heavily, because it is widely believed that DCT coefficients of middle frequencies have similar magnitudes [41]. Since the constructed system should be robust against JPEG compression, we choose the DCT coefficients in such a way that the quantization values associated with them in the JPEG compression algorithm are equal. According to Table 3.1 the coefficients (4,1) and (3,2) or (1,2) and (3,0) are good candidates.

One block encodes a "1," if $B_i(u_1, v_1) > B_i(u_2, v_2)$, otherwise a "0." In the encoding step, the two coefficients are swapped if their relative size does not match with the bit to be encoded. Since the JPEG compression can (in the quantiza-

Algorithm 3.9 DCT–Steg decoding process

for $i = 1, \ldots, \ell(M)$ **do**
  get cover-block b_i associated with bit i
  $B_i = \mathcal{D}\{b_i\}$
  if $B_i(u_1, v_1) \leq B_i(u_2, v_2)$ **then**
    $m_i = 0$
  else
    $m_i = 1$
  end if
end for

tion step) affect the relative sizes of the coefficients, the algorithm ensures that $|B_i(u_1, v_1) - B_i(u_2, v_2)| > x$ for some $x > 0$, by adding random values to both coefficients. The higher x is, the more robust the algorithm will be against JPEG compression, however, at the expense of image quality. The sender then performs an inverse DCT to map the coefficients back into the space domain. To decode the picture, all available blocks are DCT-transformed. By comparing the two coefficients of every block, the information can be restored. Embedding and extraction algorithms are outlined in Algorithms 3.8 and 3.9.

If the constant x and the location of the used DCT coefficients are chosen properly, the embedding process will not degenerate the cover visibly. We can expect this method to be robust against JPEG compression, since in the quantization process both coefficients are divided by the same quantization values. Their relative size will therefore only be affected in the rounding step.

Perhaps the most important drawback of the system presented above is the fact that Algorithm 3.8 does not discard image blocks where the desired relation of the DCT coefficients cannot be enforced without severely damaging the image data contained in this specific block.

Zhao and Koch [25, 31] proposed a similar system which does not suffer from this drawback. They operate on quantized DCT coefficients and use the relations of three coefficients in a block to store the information. The sender DCT transforms the image block b_i and performs a quantization step to get B_i^Q. One block encodes a "1," if $B_i^Q(u_1, v_1) > B_i^Q(u_3, v_3) + D$ and $B_i^Q(u_2, v_2) > B_i^Q(u_3, v_3) + D$. On the other hand, a "0" is encoded, if $B_i^Q(u_1, v_1) + D < B_i^Q(u_3, v_3)$ and $B_i^Q(u_2, v_2) + D < B_i^Q(u_3, v_3)$. The parameter D accounts for the minimum distance between two coefficients for representing an embedded bit; normally $D = 1$. The higher D is, the more robust the method will be against image processing techniques. Again, the three selected coefficients should be situated in the middle of the spectrum.

In the encoding step, the relations between these three coefficients are changed

so that they represent one bit of the secret information. If the modifications required to code one secret bit are too large, then the block is not used for information transfer and marked as "invalid." This is the case, if the difference between the largest and the smallest coefficient is greater than some constant value *MD*. The higher *MD* is, the more blocks can be used for communication. In order to allow a correct decoding, the quantized DCT coefficients of an invalid block are changed so that they fulfill one of the two conditions

$$B_i^Q(u_1, v_1) \leq B_i^Q(u_3, v_3) \leq B_i^Q(u_2, v_2) \tag{3.9}$$

or

$$B_i^Q(u_2, v_2) \leq B_i^Q(u_3, v_3) \leq B_i^Q(u_1, v_1) \tag{3.10}$$

Afterwards the block is dequantized and the inverse DCT is applied.

The receiver can restore the information by applying DCT and quantizing the block. If the three selected coefficients fulfill one of the conditions (3.9) or (3.10), the block is ignored. Otherwise the encoded information can be restored by comparing $B_i^Q(u_1, v_1)$, $B_i^Q(u_2, v_2)$, and $B_i^Q(u_3, v_3)$. The authors claim that this embedding method is robust against JPEG compression (with quality factors as low as 50%), since all changes are made after the "lossy" quantization step.

3.3.2 Hiding information in digital sound: phase coding

Embedding secret messages in digital sound is generally more difficult than embedding information in digital images. Moore [42] noted that the human auditory system is extremely sensitive; perturbations in a sound file can be detected as low as one part in 10 million. Although the limit of perceptible noise increases as the noise level of the cover increases, the maximum allowable noise level is generally quite low. It is however known that the human auditory system is much less sensitive to the phase components of sound; this fact has been exploited in numerous digital audio compression systems.

In phase coding [2], a digital datum is represented by a phase shift in the phase spectrum of the carrier signal; the carrier signal c is split into a series of N short sequences, $c_i(n)$ of length $\ell(m)$, a DFT is applied, and a matrix of the phases $\phi_i(k)$ and Fourier transform magnitudes $A_i(k)$ is created. Recall that

$$A_i(k) = \sqrt{\mathrm{Re}[\mathcal{F}\{c_i\}(k)]^2 + \mathrm{Im}[\mathcal{F}\{c_i\}(k)]^2} \tag{3.11}$$

and

$$\phi_i(k) = \arctan \frac{\mathrm{Im}[\mathcal{F}\{c_i\}(k)]}{\mathrm{Re}[\mathcal{F}\{c_i\}(k)]} \tag{3.12}$$

Since phase shifts between consecutive signal segments can easily be detected, their phase differences need to be preserved in the stego-signal. The embedding process thus inserts a secret message only in the phase vector of the first signal segment:

$$\tilde{\phi}_0(k) = \begin{cases} \pi/2 & \text{if } m_k = 0 \\ -\pi/2 & \text{if } m_k = 1 \end{cases} \tag{3.13}$$

and creates a new phase matrix using the original phase differences

$$\begin{aligned} \tilde{\phi}_1(k) &= \tilde{\phi}_0(k) + [\phi_1(k) - \phi_0(k)] \\ &\dots \\ \tilde{\phi}_N(k) &= \tilde{\phi}_{N-1}(k) + [\phi_N(k) - \phi_{N-1}(k)] \end{aligned} \tag{3.14}$$

The sender then uses the new phase matrix $\tilde{\phi}_i(k)$ and the original matrix of Fourier transform magnitudes $A_i(k)$ to construct the stego-signal using the inverse Fourier transform. Since $\phi_0(k)$ is modified, the absolute phases of all following segments are changed, while their relative differences are preserved. Before the secret information can be restored, some sort of synchronization must take place. Given the knowledge of the sequence length $\ell(m)$, the receiver is able to calculate the DFT and to detect the phases $\phi_0(k)$.

3.3.3 Echo hiding

Echo hiding [4] attempts to hide information in a discrete signal $f(t)$ by introducing an echo $f(t - \Delta t)$ in the stego-signal $c(t)$:

$$c(t) = f(t) + \alpha f(t - \Delta t) \tag{3.15}$$

Information is encoded in the signal by modifying the delay Δt between the signal and the echo. In the encoding step, the sender chooses either Δt or $\Delta t'$; in the first case, a "0" is encoded in the signal $c(t)$, in the latter case a "1." The delay times Δt or $\Delta t'$ are chosen in a way that the echo signal is not audible for a human observer.

The basic echo hiding scheme can only embed one bit in a signal; therefore a cover signal is divided into $\ell(m)$ blocks prior to the encoding process. Consecutive blocks should be separated by a random number of unused samples so that the detection and extraction of the secret message bits is harder. In each block one secret bit is embedded according to (3.15); in the last step all signal blocks are concatenated.

Before the secret message can be extracted out of the stego-signal, some sort of synchronization must take place; the receiver must be able to reconstruct the $\ell(m)$ signal blocks the sender used to embed one secret message bit. Each signal

segment can then be decoded via the autocorrelation function of the signal's cepstrum. Gruhl et al. [4] show that the autocorrelation function shows a spike at the delay time Δt. For a further investigation of echo hiding see Section 7.5.1.

Chang and Moskowitz [43] analyze several methods usable for information hiding in digital sound, among them low-bit coding (LSB), phase coding, spread spectrum techniques (see Section 3.4), and echo hiding. Low-bit coding techniques are not robust, but have the highest data transmission rate. Phase coding provides robustness against resampling of the carrier signal, but has a very low data transmission rate since secret information is encoded only in the first signal segment. On the contrary, spread spectrum and echo hiding perform better in many cases.

3.3.4 Information hiding and data compression

In some cases, information hiding algorithms are incorporated in data compression systems; one can think of a videoconferencing system which allows messages to be hidden in the video stream while it is being recorded. Most research work focussed on information hiding schemes for lossy video or image compression systems, but it should be noted that lossless compression systems can also be used for secret information transfer; Cachin [44] showed how to construct an asymptotically optimal steganographic system by modifying Willems [45] "repetition times" compression algorithm.

Numerous steganographic systems for compressed video or images have been proposed. In the simplest technique, applied by the tool Jpeg-Jsteg [46], information is hidden in the way DCT coefficients in the JPEG compression system (see Section 3.3) are rounded. Since the DCT normally outputs noninteger sequences for integer inputs, the JPEG system must quantize DCT coefficients before the encoding process. Information is hidden by rounding the coefficients either up or down according to the secret message bits. Although such a system is not robust, detection of the cover modifications seems to be difficult. Westfeld and Wolf [47] describe a similar technique. Their system operates on quantized, DCT-encoded blocks of video frames. After distinguishing blocks which are suitable for secret transmission from unusable blocks, the modulo-2 sum of the DCT coefficients of the block is changed in a way that it transmits secret information (see [47] for details). More sophisticated methods combine video compression schemes with spread spectrum. As an example, Hartung and Girod [48, 49] presented an information-hiding scheme operating on precompressed video using their spread spectrum watermarking system (see Section 6.4.1).

3.4 SPREAD SPECTRUM AND INFORMATION HIDING

Spread spectrum (SS) communication technologies have been developed since the 1950s in an attempt to provide means of low-probability-of-intercept and antijamming communications. Pickholtz et al. [50] define spread spectrum techniques as "means of transmission in which the signal occupies a bandwidth in excess of the minimum necessary to send the information; the band spread is accomplished by means of a code which is independent of the data, and a synchronized reception with the code at the receiver is used for despreading and subsequent data recovery." Although the power of the signal to be transmitted can be large, the signal-to-noise ratio in every frequency band will be small. Even if parts of the signal could be removed in several frequency bands, enough information should be present in the other bands to recover the signal. Thus, SS makes it difficult to detect and/or remove a signal. This situation is very similar to a steganography system which tries to spread a secret message over a cover in order to make it impossible to perceive. Since spreaded signals tend to be difficult to remove, embedding methods based on SS should provide a considerable level of robustness. Since the landmark paper by Tirkel et al. [51], spread spectrum methods are of increasing importance in the field of information hiding.

In information hiding, two special variants of SS are generally used: *direct-sequence* and *frequency-hopping* schemes. In direct-sequence schemes, the secret signal is spread by a constant called chip rate, modulated with a pseudorandom signal and added to the cover. On the other hand, in frequency-hopping schemes the frequency of the carrier signal is altered in a way that it hops rapidly from one frequency to the another. SS are widely used in the context of watermarking, as will be shown in Section 6.4.1. One particularly interesting direct-sequence watermarking algorithm, invented by Hartung and Girod [48, 49], which could also be used for steganographic purposes, will be described in Section 6.4.1.

Due to the similarity of SS watermarking and steganography algorithms, we will limit the discussion in this chapter to presenting a mathematical model describing the application of spread spectrum techniques in information hiding and discuss a system called SSIS as a case study.

3.4.1 A spread spectrum model

Smith and Comiskey [52] presented a general framework for spread spectrum steganography. Their approach originally used $N \times M$ grayscale images as covers; however, the work can easily be extended to all cover sets on which a scalar product can be defined. We will assume that Alice and Bob share a set of (at

least) $\ell(m)$ orthogonal $N \times M$ images ϕ_i as a stego-key. Alice first generates a stego-message $E(x, y)$ by building the weighted sum

$$E(x, y) = \sum_i m_i \phi_i(x, y) \tag{3.16}$$

The images ϕ_i are orthogonal to each other,

$$\langle \phi_i, \phi_j \rangle = \sum_{x=1}^{N} \sum_{y=1}^{M} \phi_i(x, y)\phi_j(x, y) = G_i \delta_{i,j} \tag{3.17}$$

where $G_i = \sum_{x=1}^{N} \sum_{y=1}^{M} \phi_i^2(x, y)$ and $\delta_{i,j}$ is the Kronecker delta function. Alice then encodes the secret information E in a cover C by building the element-wise sum of both images, creating the stego-cover S:

$$S(x, y) = C(x, y) + E(x, y) \tag{3.18}$$

In the ideal case, C is orthogonal to all ϕ_i, (so $\langle C, \phi_i \rangle = 0$) and Bob can extract the ith message bit m_i by projecting the stego-image S onto the ith basis image ϕ_i:

$$\begin{aligned} \langle S, \phi_i \rangle &= \langle C, \phi_i \rangle + \left\langle \sum_j m_j \phi_j, \phi_i \right\rangle \\ &= \sum_j m_j \langle \phi_j, \phi_i \rangle \\ &= G_i m_i \end{aligned} \tag{3.19}$$

Therefore, the secret information can be recovered by calculating $m_i = \langle S, \phi_i \rangle / G_i$. Note that the original cover C is not needed in the decoding phase. In practice, however, C will not be completely orthogonal to all images ϕ_i, so an error term $\langle C, \phi_i \rangle = \Delta C_i$ has to be introduced in (3.19):

$$\langle S, \phi_i \rangle = \Delta C_i + G_i m_i \tag{3.20}$$

We will now show that under reasonable assumptions the expected value of ΔC_i is zero. Let both C and ϕ_i be two independent NM-dimensional random variables. If we assume that all basis images were created using a zero-mean random process and they are independent from the messages to be transmitted, then

$$\mathbf{E}[\Delta C_i] = \sum_{i=1}^{N} \sum_{j=1}^{M} \mathbf{E}[C(x, y)]\mathbf{E}[\phi_i(x, y)] = 0 \tag{3.21}$$

Thus, the expected value of the error term in (3.20) is zero under these assumptions.

The decoding operation therefore consists of reconstructing a secret message by projecting the stego-image S onto all functions ϕ_i yielding an approximative value

$$s_i = \langle S, \phi_i \rangle = \Delta C_i + G_i m_i \tag{3.22}$$

Subject to the conditions stated above, the expected value of ΔC_i is zero, so $s_i \approx G_i m_i$. The final task is to reconstruct m_i from s_i. If we encode secret messages as strings of -1 and 1 instead of simply using binary strings, the values of m_i can be reconstructed using the *sign* function, provided that $G_i \gg 0$:

$$m_i = sign(s_i) = \begin{cases} -1 & if\ s_i < 0 \\ 0 & if\ s_i = 0 \\ 1 & if\ s_i > 0 \end{cases} \tag{3.23}$$

In the case of $m_i = 0$ the encoded information has been lost. In some severe circumstances the quantity $|\Delta C_i|$ could become so large (recall that we have only proved that the expected value is zero) that the recovery of one bit is not possible. However, this case will not happen often and can be coped with by the implementation of an error-correcting code.

The main advantage of using spread spectrum techniques in steganography is the relative robustness to image modifications. Since the encoded information is spread over a wide frequency band it is quite difficult to remove it completely without entirely destroying the cover. In practice, modifications of the stego-cover will increase the value of ΔC_i. These modifications will not be harmful to the embedded message, unless $|\Delta C_i| > |G_i m_i|$.

3.4.2 SSIS: a case study

Marvel et al. [53] presented a steganographic system called SSIS which we will discuss here briefly as a case study. SSIS uses a spread spectrum technique as an embedding function; this mechanism can be described as follows. Before the embedding process, the secret message is encrypted using a conventional symmetric encryption scheme, thereby using a secret key k_1. Furthermore, the encrypted secret message will be encoded via a low-rate error-correcting code (such as a Reed-Solomon code). This step will increase the robustness of the overall steganographic application. The resulting encoded message is then modulated by a pseudorandom sequence produced by a pseudorandom number generator using k_2 as seed. The resulting (random-looking) signal is then input into an interleaver (which uses k_3 as seed) and added to the cover. In a last step, the resulting stego-image is appropriately quantized.

At the receiver side the embedding process is reversed. Since one design goal of SSIS was to provide a *blind* steganographic system—thus, a system in which

the original image is not needed in the decoding process—an estimate of the original image is obtained using an image-restoration technique such as an adaptive Wiener filter. Subtracting the stego-image from the cover-image estimate yields an estimate for the modulated and spread stego-message. The resulting bits are then deinterleaved and demodulated (using k_3 and k_2). Due to the poor performance of the Wiener filter, the reconstructed secret message will contain incorrect bits; the stego-system can thus be seen as a form of transmission on a noisy channel. However, the use of an error-correcting code can help to recover corrupted message bits. In a last step, the secret message is decrypted.

3.5 STATISTICAL STEGANOGRAPHY

Statistical steganography techniques utilize the existence of "1-bit" steganographic schemes, which embed one bit of information in a digital carrier. This is done by modifying the cover in such a way that some statistical characteristics change significantly if a "1" is transmitted. Otherwise the cover is left unchanged. So the receiver must be able to distinguish unmodified covers from modified ones.

In order to construct a $\ell(m)$-bit stego-system from multiple "1-bit" stego-systems, a cover is divided into $\ell(m)$ disjoint blocks $B_1, \ldots, B_{\ell(m)}$. A secret bit, m_i, is inserted into the ith block by placing a "1" into B_i if $m_i = 1$. Otherwise, the block is not changed in the embedding process. The detection of a specific bit is done via a test function which distinguishes modified blocks from unmodified blocks:

$$f(B_i) = \begin{cases} 1 & \text{block } B_i \text{ was modified in the embedding process} \\ 0 & \text{otherwise} \end{cases} \tag{3.24}$$

The function f can be interpreted as a hypothesis-testing function; we test the null-hypothesis "block B_i was not modified" against the alternative hypothesis "block B_i was modified." Therefore, we call the whole class of such steganography systems *statistical steganography*. The receiver successively applies f to all cover-blocks B_i in order to restore every bit of the secret message.

The main question which remains to be solved is how such a function f in (3.24) can be constructed. If we interpret f as a hypothesis-testing function, we can use the theory of hypothesis testing from mathematical statistics. Let us assume we can find a formula $h(B_i)$, which depends on some elements of the cover-block B_i, and we know the distribution of $h(B_i)$ in the unmodified block (i.e., the hypothesis holds in this case). We can then use standard procedures to test if $h(B_i)$ equals or exceeds a specific value. If we manage to alter $h(B_i)$ in the embedding process in a way that its expected value is 0 if the block B_i was not modified, and its expected

value is much greater otherwise, we could test whether $h(B_i)$ equals zero under the given distribution of $h(B_i)$.

Statistical steganographic techniques are, however, difficult to apply in many cases. First, a good test statistic $h(B_i)$ must be found which allows distinction between modified and unmodified cover-blocks. Additionally, the distribution of $h(B_i)$ must be known for a "normal" cover; in most cases, this is quite a difficult task. In practical implementations many (quite questionable) assumptions are made in order to determine a closed formula for this distribution.

As an example, we want to construct a statistical steganography algorithm out of Pitas' watermarking system [54], which is similar to the Patchwork approach of Bender et al. [2]. Suppose every cover-block B_i is a rectangular set of pixels $p^{(i)}_{n,m}$. Furthermore, let $S = \{s^{(i)}_{n,m}\}$ be a rectangular pseudorandom binary pattern of equal size, where the number of ones in S equals the number of zeros. We will assume that both the sender and receiver have access to S, which represents the stego-key in this application. The sender first splits the image block B_i into two sets, C_i and D_i, of equal size (i.e., he puts all pixels with indices (n, m) into set C where the corresponding key bit $s_{n,m}$ equals zero):

$$\begin{aligned} C_i &= \{p^{(i)}_{n,m} \in B_i | s_{n,m} = 1\} \\ D_i &= \{p^{(i)}_{n,m} \in B_i | s_{n,m} = 0\} \end{aligned} \tag{3.25}$$

The sender then adds a value $k > 0$ to all pixels in the subset C_i but leaves all pixels in D_i unchanged. In the last step, C_i and D_i are merged to form the marked image block $\tilde{B}_i$.

In order to extract the mark, the receiver reconstructs the sets C_i and D_i. If the block contains a mark, all values in C_i will be larger than the corresponding values in the embedding step; thus we test the difference of the means of sets C_i and D_i. If we assume that all pixels in both C_i and D_i are independent identically distributed random variables with an arbitrary distribution, the test statistic

$$q_i = \frac{\overline{C_i} - \overline{D_i}}{\hat{\sigma}_i} \tag{3.26}$$

with

$$\hat{\sigma}_i = \sqrt{\frac{\mathrm{Var}[C_i] + \mathrm{Var}[D_i]}{|S|/2}} \tag{3.27}$$

where $\overline{C_i}$ denotes the mean over all pixels in the set C_i and $\mathrm{Var}[C_i]$ the estimated variance of the random variables in C_i, will follow a $N(0, 1)$ normal distribution asymptotically due to the central limit theorem. If a mark is embedded in the

image block $\tilde{B}_i$, the expected value of q will be greater than zero. The receiver is thus able to reconstruct the ith secret message bit by testing whether the statistic q_i of block B_i equals zero under the $N(0,1)$ distribution.

3.6 DISTORTION TECHNIQUES

In contrast to substitution systems, distortion techniques require the knowledge of the original cover in the decoding process. Alice applies a sequence of modifications to a cover in order to get a stego-object; she chooses this sequence of modifications in such a way that it corresponds to a specific secret message she wants to transmit. Bob measures the differences to the original cover in order to reconstruct the sequence of modifications applied by Alice, which corresponds to the secret message.

In many applications, such systems are not useful, since the receiver must have access to the original covers. If Wendy also has access to them, she can easily detect the cover modifications and has evidence for a secret communication. If the embedding and extraction functions are public and do not depend on a stego-key, it is also possible for Wendy to reconstruct secret messages entirely. Throughout this section we will therefore assume that original covers can be distributed through a secure channel.

An early approach to hiding information is in text. Most text-based hiding methods are of distortion type (i.e., the arrangement of words or the layout of a document may reveal information). One technique is by modulating the positions of lines and words, which will be detailed in the next subsection. Adding spaces and "invisible" characters to text provides a method to pass hidden information. HTML files are good candidates for including extra spaces, tabs, and linebreaks. Web browsers ignore these "extra" spaces and lines, and they go unnoticed until the source of the Web page is revealed.

3.6.1 Encoding information in formatted text

Considerable effort has been made to construct data-embedding methods for formatted text, which is interpreted as a binary image. Maxemchuk et al. [55–58] presented text-based steganographic schemes which use the distance between consecutive lines of text or between consecutive words to transmit secret information. It should be noted, however, that any steganographic system which uses the text format to transmit information can easily be broken by retyping the document.

In *line-space* encoding, the positions of lines in the document are moved up or down according to secret message bits, whereas other lines are kept stationary for the purpose of synchronization (in the original implementation, information was transmitted in every second line). One secret message bit is encoded in one line that

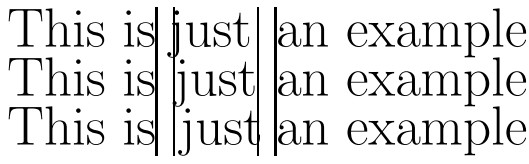

Figure 3.3 Encoding information in interword spaces (the vertical lines are provided for reference). Data is embedded in the first and third sentences.

is moved; if a line is moved up, a 1 is encoded, otherwise a 0. When decoding a secret message, *centroid detection* can be used; the centroid is defined to be the center of mass of the line about a horizontal axis. Let us denote with Δ_{R+} the distance between the centroids of a shifted line and the next stationary synchronization line above, with Δ_{R-} the distance of centroids between the shifted line and the next stationary line below, and with Δ_{X+} and Δ_{X-} the corresponding centroid distances in the unmodified document. The distance above one line was increased, if

$$\frac{\Delta_{R+} + \Delta_{R-}}{\Delta_{R+} - \Delta_{R-}} > \frac{\Delta_{X+} + \Delta_{X-}}{\Delta_{X+} - \Delta_{X-}} \tag{3.28}$$

Similarly, if

$$\frac{\Delta_{R+} + \Delta_{R-}}{\Delta_{R+} - \Delta_{R-}} < \frac{\Delta_{X+} + \Delta_{X-}}{\Delta_{X+} - \Delta_{X-}} \tag{3.29}$$

the distance above the line was decreased. Note that if the page was scaled by a constant factor during reproduction, this factor cancels out because of the fraction in (3.28) and (3.29). Similarly, changes in vertical print density should affect all centroids in approximately the same way. These properties make line-space encoding techniques resistant to most distortion attacks. For an analysis of this embedding technique see [57].

Another possible embedding scheme in formatted text is *word-space* encoding, illustrated in Figure 3.3. According to a secret message bit, horizontal spaces between selected words of the carrier are altered. Theoretically, it is possible to alter every space between two words; the only limitation is that the sum of all movements in one specific line equals zero so that the line keeps properly aligned.

3.6.2 Distortion of digital images

Distortion techniques can easily be applied to digital images. Using a similar approach as in substitution systems, the sender first chooses $\ell(m)$ different cover-pixels he wants to use for information transfer. Such a selection can again be done using pseudorandom number generators or pseudorandom permutations. To encode a 0 in one pixel, the sender leaves the pixel unchanged; to encode a 1, he adds a random

value Δx to the pixel's color. Although this approach is similar to a substitution system, there is one significant difference: the LSB of the selected color values do not necessarily equal secret message bits. In particular, no cover modifications are needed when coding a 0. Furthermore, Δx can be chosen in a way that better preserves the cover's statistical properties. The receiver compares all $\ell(m)$ selected pixels of the stego-object with the corresponding pixels of the original cover. If the ith pixel differs, the ith message bit is a 1, otherwise a 0.

Many variants of the above method could be implemented: similar to the parity bit method presented in Section 3.2.4, the parity bit of a certain image region can be altered or left unchanged in order to encode a 1 or a 0. Furthermore, image processing techniques could be applied to certain image regions so that they are not visible to an observer.

Another image distortion technique, *data embedding*, has been introduced by Sandford et al. [59, 60]. In contrast to all distortion techniques discussed so far, data embedding tries to modify the order of appearance of redundant data in the cover rather than to change values themselves; the embedding process therefore maintains a "pair list" (i.e., a list of pairs of samples whose difference is smaller than a specific threshold). The receiver can reverse the embedding process if he has access to the pair list. This list can be seen as an analogon to a key in cryptography; it normally cannot be restored out of the cover by the receiver (see [59] for details).

3.7 COVER GENERATION TECHNIQUES

In contrast to all embedding methods presented above, where secret information is added to a specific cover by applying an embedding algorithm, some steganographic applications generate a digital object only for the purpose of being a cover for secret communication.

3.7.1 Mimic functions

Due to the explosion of information traffic it can be assumed that it is impossible for a human being to observe all communications around the world; as noted in the conclusion of Chapter 2, such a task can only be done using automated supervision systems which are therefore of increasing importance. These systems check communication by examining keywords and the statistical profile of a message. It is possible, for instance, to distinguish unencrypted from encrypted messages automatically because of their different statistical properties. *Mimic functions*, proposed by Wayner [61], can be used to hide the identity of a message by changing its statistical profile in a way that it matches the profile of any innocent looking text.

It is well known that the English language possesses several statistical properties. For instance the distribution of characters is not uniform (see for instance the appendix of [62] for the frequency distributions of English di- and trigrams). This fact has been exploited in numerous data-compression techniques (e.g., the Huffman coding scheme [40]). Given an alphabet Σ and a probability distribution A, the Huffman coding scheme can be used to produce a minimum-redundancy compression function $f_A : \Sigma \to \{0,1\}^*$, where * denotes the Kleene-Star ($\Sigma^* = \bigcup_{i\geq 0}\{x_1 \cdots x_i | x_1, \ldots, x_i \in \Sigma\}$). A mimic function $g : \Sigma^* \to \Sigma^*$ that converts a message whose characters show a probability distribution A to a message which approximately mimics the statistical profile B, can be constructed using two Huffman compression functions:

$$g(x) = f_B^{-1}(f_A(x)) \tag{3.30}$$

Thus, the file x is first compressed using a Huffman scheme with distribution A. This process will create a file of binary strings which can be interpreted as output of a Huffman compression scheme (with distribution B) of a different file. This file can be reconstructed by applying the inverse Huffman compression function f_B^{-1} to the file of binary strings and will act as a stego-object. Since both f_A and f_B are one-to-one, the constructed mimic function will be one-to-one. Wayner showed that this function is optimal in the sense that if f_A is a theoretically optimal Huffman compression function and x is a file of random bits, then $f_A^{-1}(x)$ is the best approximation of the statistical profile A which is one-to-one.

Instead of using distributions of single characters, Huffman coding schemes can be constructed to compress n characters at one time, based on the frequency distribution of n-grams. However, the size of the compression tree created by the Huffman schemes grows exponentially with n. Wayner instead proposed to exploit the intercharacter dependencies by creating Huffman compression functions for every string t of length $n-1$ to encode probabilities for each character which may follow t in the file. A mimic function can be constructed out of the collection of these Huffman compression functions (see [61] for details).

3.7.2 Automated generation of English texts

However, mimic functions can only be used to fool machines. Since the stego-objects are created only according to statistical profiles, the semantic component is entirely ignored. To a human observer the created texts look completely meaningless and are full of grammatical and typographical errors.

To overcome this problem, the use of context-free grammars (CFG) has been proposed; for a theoretical overview of CFG see [63]. Let $G = \langle V, \Sigma, \Pi, S\rangle$ be a CFG, where V is the set of variables, Σ the set of terminal symbols, $\Pi \subseteq V \times (V \cup \Sigma)^*$

the set of productions and $S \in V$ the start symbol. The productions can be seen as a substitution rule; they convert a variable into a string containing terminal or variable symbols. A string $s \in \Sigma^*$ which is defined to be a sequence of terminal symbols is said to be generated by G (formally: $s \in L(G)$) if s can be produced successively from the start symbol S by substituting variables by sequences of terminal or variable symbols according to Π. For example, from the grammar $\langle\{S, A, B, C\}, \{\texttt{A}, \ldots, \texttt{Z}, \texttt{a} \ldots, \texttt{z}\}, \Pi, S\rangle$ with

$$\begin{aligned}\Pi = \{\ & S \to \texttt{Alice}\ B,\ S \to \texttt{Bob}\ B,\ S \to \texttt{Eve}\ B,\ S \to \texttt{I}\ A,\\ & A \to \texttt{am working},\ A \to \texttt{am lazy},\ A \to \texttt{am tired},\\ & B \to \texttt{is}\ C,\ B \to \texttt{can cook},\\ & C \to \texttt{reading},\ C \to \texttt{sleeping},\ C \to \texttt{working}\}\end{aligned}$$

the sentences `I am lazy`, `Alice is reading`, etc. can be derived. If for every string $s \in L(G)$ there exists exactly one way s can be generated from the start symbol, the grammar is said to be unambiguous.

Unambiguous grammars can be used as a steganographic tool. Wayner [61, 64] proposed an extension to the technique of mimic functions. Given a set of productions, we assign a probability to each possible production for variable V_i. In our example above, we could choose

$$\begin{aligned}\Pi = \{\ & S \to_{0.5} \texttt{Alice}\ B,\ S \to_{0.3} \texttt{Bob}\ B,\ S \to_{0.1} \texttt{Eve}\ B,\ S \to_{0.1} \texttt{I}\ A,\\ & A \to_{0.3} \texttt{am working},\ A \to_{0.4} \texttt{am lazy},\ A \to_{0.3} \texttt{am tired},\\ & B \to_{0.5} \texttt{is}\ C,\ B \to_{0.5} \texttt{can cook},\\ & C \to_{0.5} \texttt{reading},\ C \to_{0.1} \texttt{sleeping},\ C \to_{0.4} \texttt{working}\}\end{aligned}$$

Let $\Pi_{V_i} = \{\pi_{i,1}, \ldots, \pi_{i,n}\}$ be the set of all productions associated with variable V_i. The sender then constructs a Huffman compression function f_{Π_i} for every set Π_i. In Figure 3.4 possible Huffman trees for Π_S and Π_A are shown. Huffman compression functions can easily be derived out of these trees; for example the production "`Eve` B" will be encoded as 110, "A `am tired`" as 11, etc.

For steganographic purposes, the inverse Huffman compression functions will be used. In the encoding step, the sender derives one specific string out of the CFG which will act as the stego-object. Starting from the start symbol S, the leftmost variable V_i is changed by a production. This production is determined by the secret message and the Huffman compression function for Π_{V_i}. Specifically, the Huffman tree is traversed according to the next bits of the secret message until a node of the tree is reached. The start symbol is then substituted by the production which can be found at this node of the tree. This process is iterated (i.e., the leftmost variable is exchanged by a production which is determined by its Huffman tree and the next

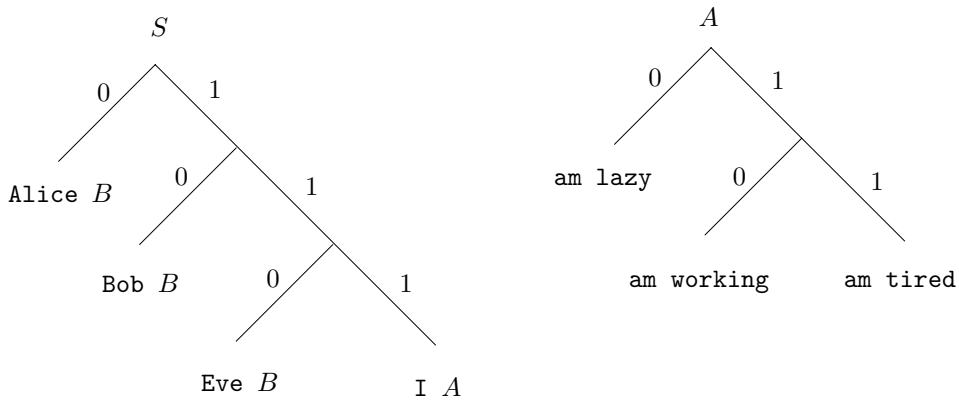

Figure 3.4 Huffman compression functions for Π_S and Π_A.

few message bits), until all message bits are used and the string consists only of terminal symbols. Continuing the previous example, suppose the secret message is 11110. In the first step we traverse the Huffman tree for Π_S and eventually reach the node "`I` A" by consuming the first three secret message bits. Thus, the start symbol S is replaced by "`I` A." We now traverse the Huffman tree Π_A and find the replacement "`am working`" by consuming another two secret message bits. Thus, since the derived string consists of terminal symbols only and all secret message bits were used, the stego-object representing 11110 is the sentence `I am working`.

In the decoding process, the cover is parsed in order to reconstruct the productions which have been used in the embedding step; this can be accomplished by the use of a parse tree for the given CFG, see [65]. Since the productions uniquely determine the secret message and the underlying grammar is unambiguous, the receiver is able to reconstruct the stego-message.

A similar system has been proposed by Chapman and Davida [66]. Their system consists of two functions, *NICETEXT* and *SCRAMBLE*. Given a large dictionary of words, categorized by different types, and a *style source*, which describes how words of different types can be used to form a meaningful sentence, *NICETEXT* transforms secret message bits into sentences by selecting words out of the dictionary which conform to a sentence structure given in the style source. *SCRAMBLE* reconstructs the secret message if the dictionary which has been used is known. Style sources can either be created from sample natural-language sentences or be generated using CFG.

3.8 CONCLUSION

In this chapter we gave an overview of different steganographic methods which have been proposed in the literature during the last few years. Many flexible and simple methods exist for embedding information in noisy communication channels.

However, covers and messages tend to have unique patterns a steganalyst could exploit. Most of the simple techniques can be broken by careful analysis of the statistical properties of the channel's noise. Images and many other signals were subject to quantization, filters, transformations, format converters, etc. Most of these techniques left some sort of "fingerprints" in the data. All these problems must be addressed when designing a steganographic system; methods which use these properties to break secret communication will be outlined in the next chapter.

REFERENCES

[1] Foley, J., et al., *Computer Graphics, Principles and Practice*, Reading, MA: Addison Wesley, 1990.

[2] Bender, W., D. Gruhl, and N. Morimoto, "Techniques for data hiding," *IBM Systems Journal*, vol. 35, no. 3/4, 1996, pp. 131–336.

[3] Möller, S., A. Pfitzmann, and I. Stirand, "Computer Based Steganography: How It Works and Why Therefore Any Restrictions on Cryptography Are Nonsense, At Best," in *Information Hiding: First International Workshop, Proceedings*, vol. 1174 of *Lecture Notes in Computer Science*, Springer, 1996, pp. 7–21.

[4] Gruhl, D., A. Lu, and W. Bender, "Echo Hiding," in *Information Hiding: First International Workshop, Proceedings*, vol. 1174 of *Lecture Notes in Computer Science*, Springer, 1996, pp. 295–316.

[5] Kurak, C., and J. McHughes, "A Cautionary Note On Image Downgrading," in *IEEE Computer Security Applications Conference 1992, Proceedings*, IEEE Press, 1992, pp. 153–159.

[6] van Schyndel, R. G., A. Tirkel, and C. F. Osborne, "A Digital Watermark," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 2, 1994, pp. 86–90.

[7] Johnson, N. F., and S. Jajodia, "Exploring Steganography: Seeing the Unseen," *IEEE Computer*, vol. 31, no. 2, 1998, pp. 26–34.

[8] Gerzon, M. A., and P. G. Graven, "A High-Rate Buried-Data Channel for Audio CD," *Journal of the Audio Engineering Society*, vol. 43, no. 1/2, 1995, pp. 3–22.

[9] "StegoDos—Black Wolf's Picture Encoder v0.90B," `<ftp://ftp.csua.berkeley.edu/pub/cypherpunks/steganography/stegodos.zip>`, 1993.

[10] Brown, A., "S-Tools for Windows," `<ftp://idea.sec.dsi.unimi.it/pub/security/crypt/code/s-tools4.zip>`, 1996.

[11] Hastur, H., "Mandelsteg," `<ftp://idea.sec.dsi.unimi.it/pub/security/crypt/code/steg.tar.Z>`, 1994.

[12] Machado, R., "EzStego, Stego Online, Stego," `<http://www.stego.com>`, 1997.

[13] Maroney, C., "Hide and Seek," <ftp://ftp.csua.berkeley.edu/pub/cypherpunks/steganography/hdsk41b.zip>, <http://www.rugeley.demon.co.uk/security/hdsk50.zip>, 1994–1997.

[14] Repp, H., "Hide4PGP," <http://www.rugeley.demon.co.uk/security/hide4pgp.zip>, 1996.

[15] Arachelian, R., "White Noise Storm," <ftp://ftp.csua.berkeley.edu/pub/cypherpunks/steganography/wns210.zip>, 1994.

[16] Hansmann, F., "Steganos, Deus Ex Machina Communications," <http://www.steganography.com/>, 1996.

[17] Menezes, A. J., P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, Boca Raton: CRC Press, 1996.

[18] Aura, T., "Practical Invisibility in Digital Communication," in *Information Hiding: First International Workshop, Proceedings*, vol. 1174 of *Lecture Notes in Computer Science*, Springer, 1996, pp. 265–278.

[19] Luby, M., and C. Rackoff, "How to Construct Pseudorandom Permutations from Pseudorandom Functions," *SIAM Journal on Computation*, vol. 17, no. 2, 1988, pp. 373–386.

[20] Naor, M., and O. Reingold, "On the Construction of Pseudorandom Permutations: Luby-Rackoff Revisited," *Journal of Cryptology*, vol. 12, no. 1, 1999, pp. 29–66.

[21] Bell, D. E., and L. J. LaPadula, "Secure Computer Systems: Mathematical Foundations," Mitre Report ESD-TR-73-278 (Vol. I–III), Mitre Corporation, Bedford, MA, Apr. 1974.

[22] Fridrich, J., "A New Steganographic Method for Palette-Based Images," in *Proceedings of the IS&T PICS conference*, Savannah, Georgia, Apr. 1998, pp. 285–289.

[23] Matsui, K., and K. Tanaka, "Video-Steganography: How to Secretly Embed a Signature in a Picture," *IMA Intellectual Property Project Proceedings*, vol. 1, no. 1, 1994, pp. 187–205.

[24] Baharav, Z., and D. Shaked, "Watermarking of Dither Halftoned Images," in *Proceedings of the SPIE 3657, Security and Watermarking of Multimedia Content*, 1999, pp. 307–316.

[25] Zhao, J., and E. Koch, "Embedding Robust Labels into Images for Copyright Protection," in *Proceedings of the International Conference on Intellectual Property Rights for Information, Knowledge and New Techniques*, München, Wien: Oldenbourg Verlag, 1995, pp. 242–251.

[26] "CCITT Recommendation T6: Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus for Document Transmission," 1984.

[27] Anderson, R. J., R. Needham, and A. Shamir, "The Steganographic File System," in *Proceedings of the Second International Workshop on Information Hiding*, vol. 1525 of *Lecture Notes in Computer Science*, Springer, 1998, pp. 73–82.

[28] "ScramDisk: Free Hard Drive Encryption For Windows 95 & 98," <http://www.scramdisk.clara.net>, 1998.

[29] Handel, T. G., and M. T. Sandford, "Data Hiding in the OSI Network Model," in *Information Hiding: First International Workshop, Proceedings*, vol. 1174 of *Lecture Notes in Computer Science*, Springer, 1996, pp. 23–38.

[30] Cox, I., et al., "A Secure, Robust Watermark for Multimedia," in *Information Hiding: First International Workshop, Proceedings*, vol. 1174 of *Lecture Notes in Computer Science*, Springer, 1996, pp. 185–206.

[31] Koch, E., and J. Zhao, "Towards Robust and Hidden Image Copyright Labeling," in *IEEE Workshop on Nonlinear Signal and Image Processing*, Jun. 1995, pp. 452–455.

[32] Koch, E., J. Rindfrey, and J. Zhao, "Copyright Protection for Multimedia Data," in *Proceedings of the International Conference on Digital Media and Electronic Publishing*, Leeds, UK, Dec. 1994.

[33] Ó Runaidh, J. J. K., F. M. Boland, and O. Sinnen, "Watermarking Digital Images for Copyright Protection," in *Electronic Imaging and the Visual Arts, Proceedings*, Feb. 1996.

[34] Xia, X., C. G. Boncelet, and G. R. Arce, "A Multiresolution Watermark for Digital Images," in *Proceedings of the IEEE International Conference on Image Processing (ICIP'97)*, 1997.

[35] Rhodas, G. B., "Method and Apparatus Responsive to a Code Signal Conveyed Through a Graphic Image," U.S. Patent 5,710,834, 1998.

[36] Swanson, M. D., B. Zhu, and A. H. Tewfik, "Transparent Robust Image Watermarking," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 3, 1996, pp. 211–214.

[37] Langelaar, G., J. van der Lubbe, and R. Lagendijk, "Robust Labeling Methods for Copy Protection of Images," in *Proceedings of the SPIE vol. 3022, Storage and Retrieval for Image and Video Databases V*, 1997, pp. 298–309.

[38] Pennebaker, W. B., and J. L. Mitchell, *JPEG Still Image Compression Standard*, New York: Van Nostrand Reinhold, 1993.

[39] Wallace, G. K., "The JPEG Still Picture Compression Standard," *Communications of the ACM*, vol. 34, no. 4, 1991, pp. 30–44.

[40] Huffman, D. A., "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE*, vol. 40, no. 10, 1952, pp. 1098–1101.

[41] Smoot, S., and L. A. Rowe, "DCT Coefficient Distributions," in *Proceedings of the SPIE 2657, Human Vision and Electronic Imaging*, 1996, pp. 403–411.

[42] Moore, B. C. J., *An Introduction to the Psychology of Hearing*, London: Academic Press, 1989.

[43] Chang, L., and I. S. Moskowitz, "Critical Analysis of Security in Voice Hiding Techniques," in *Proceedings of the International Conference on Information and Communications Security*, vol. 1334 of *Lecture Notes in Computer Science*, Springer, 1997, pp. 203–216.

[44] Cachin, C., "An Information-Theoretic Model for Steganography," in *Proceedings of the Second International Workshop on Information Hiding*, vol. 1525 of *Lecture Notes in Computer Science*, Springer, 1998, pp. 306–318.

[45] Willems, F. M., "Universal Data Compression and Repetition Times," *IEEE Transactions on Information Theory*, 1989, pp. 337–343.

[46] Upham, D., "Jpeg-Jsteg, modification of the independent JPEG group's JPEG software (release 4) for 1-bit steganography in JFIF output files," `<ftp://ftp.funet.fi/pub/crypt/steganography/>`, 1992–1997.

[47] Westfeld, A., and G. Wolf, "Steganography in a Video Conferencing System," in *Proceedings of the Second International Workshop on Information Hiding*, vol. 1525 of *Lecture Notes in Computer Science*, Springer, 1998, pp. 32–47.

[48] Hartung, F., and B. Girod, "Copyright Protection in Video Delivery Networks by Watermarking of Pre-Compressed Video," in *Multimedia Applications, Services and Techniques—ECMAST 97*, vol. 1242 of *Lecture Notes in Computer Science*, Springer, 1997, pp. 423–436.

[49] Hartung, F., and B. Girod, "Watermarking of Uncompressed and Compressed Video,"

Signal Processing, vol. 66, no. 3, 1998, pp. 283–301.

[50] Pickholtz, R. L., D. L. Schilling, and L. B. Milstein, "Theory of Spread-Spectrum Communications—A Tutorial," *IEEE Transactions on Communications*, vol. 30, no. 5, 1982, pp. 855–884.

[51] Tirkel, A. Z., G. A. Rankin, and R. van Schyndel, "Electronic Watermark," in *Digital Image Computing, Technology and Applications—DICTA 93, Macquarie University*, 1993, pp. 666–673.

[52] Smith, J., and B. Comiskey, "Modulation and Information Hiding in Images," in *Information Hiding: First International Workshop, Proceedings*, vol. 1174 of *Lecture Notes in Computer Science*, Springer, 1996, pp. 207–227.

[53] Marvel, L. M., C. G. Bonclet, and C. T. Retter, "Reliable Blind Information Hiding for Images," in *Proceedings of the Second International Workshop on Information Hiding*, vol. 1525 of *Lecture Notes in Computer Science*, Springer, 1998, pp. 48–61.

[54] Pitas, I., "A Method for Signature Casting on Digital Images," in *International Conference on Image Processing*, vol. 3, IEEE Press, 1996, pp. 215–218.

[55] Maxemchuk, N. F., "Electronic Document Distribution," *AT&T Technical Journal*, September/October 1994, pp. 73–80.

[56] Low, S. H., et al., "Document Marking and Identification Using Both Line and Word Shifting," in *Proceedings of Infocom'95*, 1995, pp. 853–860.

[57] Low, S. H., N. F. Maxemchuk, and A. M. Lapone, "Document Identification for Copyright Protection Using Centroid Detection," *IEEE Transactions on Communications*, vol. 46, no. 3, 1998, pp. 372–383.

[58] Low, S. H., and N. F. Maxemchuk, "Performance Comparison of Two Text Marking Methods," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, 1998, pp. 561–572.

[59] Sandford, M. T., J. N. Bradley, and T. G. Handel, "Data Embedding Method," in *Proceedings of the SPIE 2615, Integration Issues in Large Commercial Media Delivery Systems*, 1996, pp. 226–259.

[60] Sandford, M. T., T. G. Handel, and J. M. Ettinger, "Data Embedding in Degenerate Hosts," Technical Report LA-95-4446UR, Los Alamos National Laboratory, 1996.

[61] Wayner, P., "Mimic Functions," *Cryptologia*, vol. XVI/3, 1992, pp. 193–214.

[62] "Basic Cryptanalysis," Headquarters Department of the Army, Field Manual NO 34-40-2, <ftp://ftp.ox.ac.uk/cryptanalysis/basic_cryptanalysis.ps.tar.gz>.

[63] Hopcroft, J. E., and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Reading, MA: Addison Wesley, 1979.

[64] Wayner, P., "Strong Theoretical Steganography," *Cryptologia*, vol. XIX/3, 1995, pp. 285–299.

[65] Aho, A., R. Sethi, and J. Ullman, *Compilers: Principles, Techniques and Tools*, Reading (MA): Addison Wesley, 1986.

[66] Chapman, M., and G. Davida, "Hiding the Hidden: A Software System for Concealing Ciphertext as Innocuous Text," in *Proceedings of the International Conference on Information and Communications Security*, vol. 1334 of *Lecture Notes in Computer Science*, Springer, 1997, pp. 335–345.