From the Proceedings of the SPIE, 2420:40, San Jose CA, February, 1995

Techniques for data hiding

Walter Bender, Daniel Gruhl, and Norishige Morimoto
Massachusetts Institute of Technology, Media Laboratory
Cambridge, Massachusetts 02139 USA

ABSTRACT

Data hiding, or steganography, is the process of embedding data into image and audio signals. The process is constrained by the quantity of data, the need for invariance of the data under conditions where the “host” signal is subject to distortions, e.g. compression, and the degree to which the data must be immune to interception, modification, or removal. We explore both traditional and novel techniques for addressing the data hiding process and evaluate these techniques in light of three applications: copyright protecting, tamper-proofing and augmentation data embedding.

1. INTRODUCTION

Digital media facilitate access to audio and image data, potentially improving the portability, efficiency, and accuracy of the information. Side effects of facile data access are violation of copyright, and tampering or modification of content. We are investigating data hiding as a means of protecting intellectual property rights and as an indicator of content manipulation. Data hiding is a class of processes used to embed data, such as copyright information, into media such as image and audio with a minimum amount of degradation to the “host” signal, i.e., the embedded data should be invisible or inaudible to a human observer. Note that data hiding is distinct from encryption. The goal of data hiding is not to restrict or regulate access to the host signal, but rather to ensure that embedded data remains inviolate.

The primary purposes for data hiding in digital media are to provide solid proof of the copyright and assurance of content integrity. Therefore, the data should stay hidden in the host, even if the host signal is subjected to manipulation, such as filtering, re-sampling, cropping, or data compression. Other applications, such as augmentation data embedding need not be invariant to detection or removal, since these data are there for the benefit of both the author and the content consumer. Thus, the techniques used for data hiding vary depending on the quantity of data being hidden and the required invariance to manipulation. A class of processes are needed to span the entire range of the applications.

The technical challenges of data hiding are formidable. Whatever “hole” in the host signal one finds to fill with data is likely to be eliminated by signal compression. The key to successful data hiding is to find holes that are not suitable for exploitation by compression algorithms. A further challenge is to fill these holes with data that remains invariant to host signal transformations. Data hiding techniques should be capable of embedding data in a host signal with the following restrictions and features: (1) Degradation of the host signal should be minimized - the embedded data needs to be “invisible” or “inaudible”; (2) Embedded data needs to be directly encoded into the image or audio signal itself, rather than into a header or wrapper so it remains intact across varying data file formats; (3) The embedded data should be immune to modifications ranging from intentional and intelligent removal to common usage, e.g., channel noise, filtering, re-sampling, cropping, encoding, compressing, etc.; (4) Asymmetrical coding of the embedded data is desirable since the purpose of data hiding is to keep the data in the host signal, but not necessarily to make it difficult to be access; (5) It is inevitable that some degradation to the data when the host signal is modified. Error correction coding[1] is used to ensure hidden data integrity; (6) Finding hidden data in a modified host signal also presents a challenge. The embedded data should be self-clocking or arbitrarily re-entrant.

1.1 Applications

Several prospective applications of data hiding are discussed in this section. The amount of data to be hidden and the expected level of modification for each application are different, therefore, different processes based on different techniques are used for each application. There are always trade-offs between the amount of data hidden and the level of immunity to modification.

An application that requires minimal data to be embedded is a digital watermark. The embedded data are used to mark the ownership of the host signal, e.g., an author’s signature or a company logo. Since the information is critical and the signal may face intelligent and intentional destruction, the coding technique must be immune to modifications. Tamper-proofing is used to indicate that the host signal has been modified from its authored state. Modification to the hidden data indicates that the host signal has been changed. Another application, feature location, requires more data. In this application, the embedded data is hidden in correspondence to specific locations within an image. It enables one to identify the individual content features, e.g., the name of the person or a brief description of the scene. Typically, the data is not removed intentionally. However, the image can be subjected to certain degree of retouching such as scaling, cropping, and contrast enhancement. So the data hiding process needs to be immune to geometrical and non-geometrical modifications of the host signal. Image and audio captions (or annotations) may require a large amount of data. Annotations often travel separately from the host signal, requiring additional channels and storage. Annotations stored in file header or resource sections are often lost if the file format is changed, e.g., the annotations created by TIFF will not be recognized by GIF.   Annotations embedded directly in the data structure of the host signal resolves these problems.

1.2 Prior work

Adelson[2] describes a method of data hiding that exploits the human visual system’s varying sensitivity to contrast versus spatial frequency. Adelson substitutes high-spatial frequency image data for “hidden” data in a pyramid-encoded still image. While he is able to encode a large amount of data efficiently, there is no provision to make the data immune to detection or removal by typical manipulations such as filtering and re-scaling. Stego[3] simply encodes data in the least-significant bit of the host signal. This technique suffers from all of the same problems as Adelson’s method, but creates an additional problem of degrading image or audio quality. Bender[4] modifies method Adelson’s technique by using “chaos” as a means to encrypt the embedded data, deterring detection, but provides no improvement to immunity to host signal manipulation. Lippman[5] hides data in the chrominance signal of NTSC by exploiting the temporal over-sampling of color. Typical of Enhanced Definition Television Systems, the method encodes a large amount of data, but the data is lost to most recording, compression and transcoding processes. Other techniques, such as Xerox’s DataGlyph[6], which adds a “bar code” to images, are engineered in light of a predetermined set of geometric modifications[7]. Spread-spectrum[8,9], a promising technology for data hiding, is difficult to detect, but is not robust to many host signal transformations.

1.3 Problem space

The requirements for each of the proposed data hiding applications vary. Some applications require robustness while others require large amounts of data. The data hiding problem space is illustrated in Table 1. The horizontal-axis represents the amount of data to embed, and the vertical-axis represents the level of modification anticipated to be performed to the host signal. The “X” demarcates problems explored in this paper.

Table 1: The data hiding problem space

	Application	watermark	feature location/ tamper-proofing	caption
Robustness	intentional removal	X		
	non-geometric modifications kernel, compression, etc.	X		
	geometric modifications affine, cropping, etc.	X	X	
	no modifications	X	X	X
	Data quantity	small (bits)		large (kilobits)

2. DATA HIDING IN STILL IMAGES

Data hiding in still images presents a variety of interesting challenges that arise due to the statistical nature of still images and the typical modifications to which they are subject. Still images provide a relatively short host signal in which to hide data. A fairly typical 8-bit picture of 200x200 pixels provides only a 40Kbyte data space with which to work. This is equivalent to only 5 seconds of telephone quality audio or less than a single frame of television. Also, it is reasonable to expect that still images will be subject to operations ranging from simple affine transforms to cropping, blurring, filtering, and compression. Usable data hiding techniques need to be resistant to as many of these transformations as possible.

Despite these problems, still images are likely candidates for data hiding. There are many attributes of the human visual system (HVS) that are potential candidates for exploitation by data hiding, including our varying sensitivity to contrast as a function of spatial frequency and the masking effect of edges (both in luminance and chrominance). The HVS has low-sensitivity to small changes in gray-scale, approximately one part in 30 for continuous random patterns. However, in uniform regions of an image, the HVS is more sensitive to the change of the gray-level, approximately one part in 240. Since most pictures allow for one part in 256, e.g. 8-bit gray levels, there is potentially room to hide data as pseudo-random changes to picture brightness. Another HVS "hole" we exploit is our relative insensitivity to very low spatial frequencies such as continuous changes in brightness across an image, i.e., vignetting. Another advantage of working with still images is that they are non-causal. Data hiding techniques can have access to any pixel or block of pixels at random.

Using these observations, we have developed a variety of techniques for placing data in images. Some of them are more suited to dealing with small amounts of data, others to large amounts. Some techniques are highly resistant to geometric modifications and others are more resistant to non-geometric modifications, e.g., filtering.

2.1 Low bit-rate data hiding

With low bit-rate encoding, we expect a high level of robustness in return for low bandwidth. The emphasis is on resistance to attempts of data removal by a third party. Two different approaches are discussed: (1) one that uses a statistical approach, and (2) one that exploits perceptual characteristics of the HVS.

2.1.1 Patchwork: a statistical approach

The statistical approach, which we refer to as "Patchwork," is based on a pseudo-random, statistical process. This method embeds one bit of data in a host image invisibly. Since the process is independent of the contents of the host image, it shows reasonably high immunity to most image modifications.

The Patchwork algorithm proceeds as follows: Take any two points, *A* and *B*, chosen at random in an image. Let *a* equal the brightness at *A* and *b* the brightness at *B*. If we subtract *a* from *b*, the *expected* value of the subtraction is zero. Repeating this procedure *n* times, letting a_i and b_i be the *ith* values of *A* and *B*, yields the sum *S*.

$$S = \sum_{1}^{n} a_i - b_i$$

The *expect*ed value of *S* is *0* after many iterations, as the number of times a_i is greater than b_i should be offset by the number of times the reverse is true. If *S* strays too far from zero, it is safe to assume that the sample space is not completely random. Assuming that (1) we are operating in a 256 level linear quantized system starting at 0, (2) all brightness levels are equally likely, and (3) all samples are independent of all other samples, the variance for a single a_i-b_i pair is calculated to be: $2 \times \sigma_i^2 = 10922.5$.

$$2 \times \sigma_i^2 = 2 \times \sum_{0}^{255} (i - (255/2))^2 = 2 \times 5461.25 = 10922.5$$

As each subtraction is done, the variance builds up (since the samples are independent), so for *n* samples, the expected variance is: $\sigma^2 = 10922.5 \times n$. Using a statistical bounding method such as Chebyshev's inequality, it is possible to calculate the likelihood that a particular picture has occurred through random choice of points. Now, by choosing the sequence of points in the picture using a pseudo-random number generator and a known seed, the seed becomes the key. Iterate through the picture for *n* points, moving a_i up one brightness quanta, and b_i down one. This encodes the picture. To decode the picture, calculate the sum *S*. Since the expected value of the difference of two points is now *2*, the expected sum *S* will now be $2 \times n$. As *n* increases, the degree of certainty that this *S* could not have been arrived at unless the picture is encoded rapidly approaches 100%.

Using this basic method, we have encoded a number of pictures. These modifications improve performance: (1) Treat "patches" of several points rather than single points. This has the effect of shifting the noise introduced by Patchwork into lower spatial frequencies, where it is less likely to be removed by compression and FIR filters; (2) Patchwork decoding is highly sensitive to affine transformations of the host image. If the path of iteration through the picture is offset by translation, rotation, or scaling between encoding and decoding, the code is lost. A combination with either affine coding (described in section 2.2.1) or some heuristic based upon feature recognition (e.g., alignment to the intraocular line of a face) is necessary to make Patchwork more robust; (3) Patchwork is remarkably resistant to cropping. By disregarding points outside of the known picture area, Patchwork degrades in accuracy approximately as *ln* of the picture size.   Patchwork is also resistant to gamma and tone scale correction since adjacent brightness values move roughly the same way under such modifications.

To validate Patchwork, an experiment was done on three AP wire photographs, each converted to approximately 200x300 pixels and 8 bits per pixel. Our results using this method are encouraging (See Figure 1). A typical 200x300 pixel image can be encoded to between 98% and 99.9% certainty. This is resistant to kernel modifications up to a 5x5 kernel, and after JPEG compression, with parameters set to 75% quality and 0% smoothing, the picture is still encoded to 85% certainty.

The major limitation to this technique is the extremely low bandwidth, usually one bit. However, without the key for the pseudo-random number generator, it is nearly impossible to remove this coding without blurring or obscuring the picture beyond recognition.

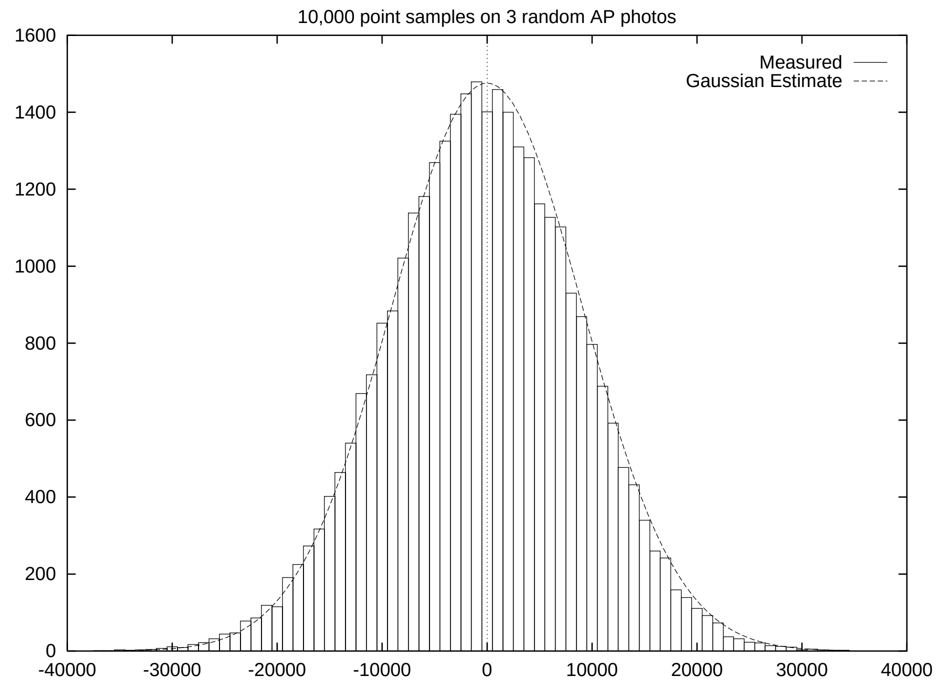

Figure 1: Experimental results from Patchwork. The sum *S* is plotted along the horizontal axis. The number of trials that yield *S* is plotted along the vertical axis. The *expected* value of *S* is 74.79 where *n*=10,000.

2.1.2 Texture Block Coding: a visual approach

Another method for low bit-rate data hiding is “Texture Block Coding.” This method hides data in continuous random texture patterns. The texture blocking scheme is implemented by copying a region from a random texture pattern found in a picture to an area which has similar texture. This results in a pair of identically textured regions in the image. The auto-correlation of the image results in the recovery of the shape of the region.

Since the two regions are identical, they are modified in the same way when if picture is uniformly transformed. By making the region reasonably large, the inner part of the region is immune to most non-geometric transformations. In our experiments, coded 16x16 pixel blocks can be decoded even when the picture is subjected to a combination of filtering, compression and rotation.

Texture block coding is not without its disadvantages. Currently it requires a human operator to choose the source and destination regions, and to evaluate the visual impact upon the image. It is possible to automate this process using a texture recognition system. However, this technique will not work on images that lack moderately large areas of random texture from which to draw.

Future research in this area includes the possibility of cutting and pasting blocks from only part of the image frequency spectrum. This would allow less noticeable blocks to be moved around, and a final image that is considerably more robust to various image compression algorithms.

2.2 High bit-rate coding

As illustrated in Table 1, high bit-rate methods tend not to be immune to image modifications. The most common form of high bit-rate encoding is simply replacing the least significant bit of the image data with the embedded data. Other techniques include the introduction of high-frequency, low-amplitude noise and direct spread-spectrum. All of the high bit-rate methods can be made more robust through the use of error-correction coding, at the expense of data rate. Consequently, high bit-rate codes are only appropriate where it is reasonable to expect that a great deal of control is maintained over the images.

Individually, none of the techniques developed are resistant to all possible transforms. In combination, often one technique can supplement another. Supplementary techniques are particularly important for recovery from geometric modifications such as affine transformations, and maintaining synchronization for spread-spectrum encoding.

2.2.1 Affine coding

Some of the data hiding techniques, such as Patchwork, are vulnerable to affine transforms. It makes sense to develop techniques that can be used in conjunction with others to provide the ability to recover data after affine application. “Affine coding” is such a technique. Any one of the high bit-rate coding techniques is used to encode pre-defined reference patterns into an image. Estimation of any geometric transformation of the image is achieved by comparing the original shape, size, and orientation of the reference patterns to those found in the transformed image. Since affine transforms are linear, the inverse transform can be applied to recover the original image. Once this is done, the image is ready for further extraction of hidden data.

2.3 Applications

The techniques outlined above for placing data in images are useful in a variety of applications. Most of the applications (digital watermark, feature location, embedded captions, and tamper proofing) are suited to one of the above techniques, depending on data requirements and anticipated modifications to the host signal.

2.3.1 Digital Watermark

The object of “Digital Watermark” is to place an indelible mark on an image. Usually, this means encoding only a handful of bits, sometimes as few as one. This has several uses, including the protection of on-line images for news services, and for photographers who are selling their work for publication. One can imagine a digital camera placing such a watermark on every photo it takes, allowing the photographer to be identified wherever the image appears.

It can be expected that if information about legal ownership is to be included in an image, it is likely that someone might want to remove it. If this is a concern then techniques used in Digital Watermark need to be hard to remove. Both the Patchwork and Texture Block Coding techniques show promise for Digital Watermark, with Patchwork used for a more secure system, and Texture Block Coding for a system the public can access.

2.3.2 Feature location

Another application of data hiding is feature location. Using data hiding it is possible for an editor (or machine) to encode descriptive information, such as the location and identification of features of interest, directly into an image. This enables retrieval of the descriptive information. Since the information is spatially located in the image, it is not removed unless the feature of interest is removed. It also translates, scales and rotates exactly as the feature of interest does.

This application does not have the same requirements for robustness as the digital watermark. It can be assumed that since the feature location is providing a service, it is unlikely someone will maliciously try to remove the encoded information.

2.3.3 Embedded captions

Typical news photograph captions are approximately one kilobyte of data. Thus embedding captions is a relatively high bit-rate application for data hiding. Like feature location, caption data is usually immune to malicious removal. While captions are useful by themselves, they become even more useful when combined with feature location. It is then possible for portions of the caption to directly reference items in the picture. Captions can self-edit once this is done. If an item referenced in the caption is cropped out of the picture, then the reference to that item in the caption can be removed automatically.

2.3.4 Tamper Proofing

Both the Digital Watermark and the Tamper Proofing applications pronounce a one bit judgement. Digital Watermark answers the question; “Is the image owned by someone?” Tamper Proofing answers the question; “Has this image been modified?”

There are several ways to implement Tamper Proofing. The easiest way is to encode a check-sum of the image within the image. It is useful to consider a Tamper Proofing algorithm that is not triggered by small changes in the image, such as cropping and gamma correction, but is triggered by gross changes, such as removing or inserting items or people. This suggests an approach involving a pattern overlaid on the image. The key to a successful overlay is to find a pattern resistant to filtering and gamma correction, yet is not be removed easily. This problem remains an active area of research.

3. DATA HIDING IN AUDIO

Data hiding in audio signals provides a special challenge because the human auditory system (HAS) is extremely sensitive. The HAS is sensitive to a dynamic range of amplitude of one billion to one and of frequency of one thousand to one. Sensitivity to additive random noise is also acute. The pertibations in a sound file can be detected as low as one part in ten million (-80dB). Although the limit of perceptible noise increases as the noise contents of the host audio signal increases, the typical allowable noise level is very low. The HAS has very low sensitivity to the phase of the sound. Unfortunately, this “hole” has been exploited by numerous audio compression algorithms.

3.1 Audio environments

There are several environments that need to be considered when hiding data in an audio host signal. An end-to-end digital audio environment has the advantage of absolute amplitude, phase and temporal quantization. Host audio signals that pass through an analog stage and are subsequently re-digitized have only relative characteristic values. Signals that stay digitally encoded but undergo re-sampling may preserve amplitude and phase accurately, but have changing temporal quantization.

Currently, the most popular format for high quality audio is in 16-bit quantization, e.g. WAV on PCs and AIFF on Macintosh^{TM} (8-bit μ-law is also popular). 16-bit quantization yields quanta that is only one part in 65536, approximately 150 times larger than the minimum perceptual level. This would suggest that random noise is not the best way to proceed. One possibility is to adapt the noise insertion to the host signal content. Another is to modify the phase of each frequency component of the sound. This would be ideal to hide data inaudibly, except that many audio compression algorithms, e.g. the ISO MPEG-AUDIO standard, cause considerable corruption of phase information. As is the case in data hiding with still images, it is necessary to investigate several methods for audio, with the understanding that there is no universal answer to “what is the best."

One additional challenge to data hiding in audio is the likelihood that the host audio signal is transfered to analog during its transmission or storage stage. Digital-to-analog conversion introduces noise and distortions to the waveform.

3.2 Low-bit coding

Low-bit coding is the simplest way to embed data into other data structures. By replacing the least significant bit of each sampling point by a coded binary string, we can code a large amount of data in an audio signal. Ideally, the channel capacity will be 8kbps in an 8kHz sampled sequence and 44kbps in a 44kHz sampled sequence for an noiseless channel application. In return for this large channel capacity, audible noise is introduced. The impact of this noise is a direct function of the content of the host signal, e.g., a live sports event contains crowd noise that masks the noise resultant from low-bit encoding.

The major disadvantage of this method is its poor immunity to manipulation. Encoded information can be destroyed by channel noise, re-sampling, etc., unless it is coded using redundancy techniques, which reduces the data rate one to two orders of magnitude. In practice, it is useful only in closed, digital-to-digital environments.

3.3 Phase coding

Phase coding, when it can be used, is one of the most effective coding schemes in terms of the signal-to-noise ratio. When the phase relation between each frequency components is dramatically changed, phase dispersion and “rain barrel” distortions occur. However, as long as the modification of the phase is within certain limits an “inaudible” coding can be achieved (See section 3.3.2).

3.3.1 Procedure

The procedure for the phase coding is as follows:
(1) Break the sound sequence *s[n]* into a series of *M* short segments, $s_i[n]$;
(2) Apply a *N*-points Discrete Short Time Fourier Transform (STFT)[10] to *i-th* segment, $s_i[n]$, and create a matrix of the phase, $\{\phi_i(\omega_k)\}$,and Fourier transform magnitude, $\{A_i(\omega_k)\}$ for $(1 \le k \le N)$, where ϕ_i denotes the phase and A_i the magnitude corresponding to frequency ω_k;
(3) Store the phase difference between each adjacent segment for $(0 \le i \le M-1)$:

$$\Delta\phi_i(\omega_k) = \phi_{i+1}(\omega_k) - \phi_i(\omega_k)$$

(4) The binary string used to modify the phase can be a series of a code words. Add these codes to the first set of entries in the phase matrix:

$$\hat{\phi}_0(\omega_k) = \phi_0(\omega_k) + d_n$$

(5) Re-create phase matrixes for $i > 0$ by using the phase difference:

$$(\hat{\phi}_1(\omega_k) = \hat{\phi}_0(\omega_k) + \Delta\hat{\phi}_1(\omega_k))$$
$$\dots$$
$$(\hat{\phi}_i(\omega_k) = \hat{\phi}_{i-1}(\omega_k) + \Delta\hat{\phi}_i(\omega_k))$$

(6) Use the modified phase matrix $\phi_i(\omega_k)$ and the original Fourier transform magnitude $A_i(\omega_k)$ to reconstruct the sound signal by applying the inverse STFT.

For the decoding process, the synchronization of the sequence is done before the decoding. The length of the segment, the DFT points, and the data interval must be known at the receiver. The value of the underlying phase of the first segment is detected as a 0 or 1, which represents the coded binary string.

Since $\phi_0(\omega_k)$ is modified, the absolute phases of the following segments are modified respectively. However, the relative phase difference of each adjacent frame are preserved. The reconstructed waveform is close enough to the original waveform to be indistinguishable.

3.3.2 Evaluation

Phase dispersion is a distortion caused by a break in the relationship of the phases between each of the frequency components. Minimizing phase dispersion constrains the data rate of phase coding. One cause of phase dispersion is the substitution of phase $\phi_0(\omega_k)$ with the binary code. To minimize error, the magnitude of the phase modifier needs to be as close as possible to the original value. To minimize noise susceptibility the difference between phase modifier states should be maximized. Phase ranges from $-\pi$ to $+\pi$. Our modified phase ranges from 0 to 1.

Another source of distortion is the rate of change of the phase modifier. If the change of the value is as often as one frequency slot per bit, it is likely to break the phase relationship of the adjacent frequency components. Replicating neighbor frequency slots together minimizes audible distortions in reconstruction. Replication causes a linear reduction in the data rate.

As a result of the examination of sound contexts (see section 3.5), we conclude that, for host sounds with quiet backgrounds, a channel capacity of 8bps can be achieved by allocating 128 frequency slots per bit. For host sounds with noisy backgrounds, 16bps to 32bps can be achieved by allocating 32 to 64 frequency slots per slot.

3.4 Spread Spectrum

There is an interesting parallel between data hiding in audio and the work on spread spectrum radio communication. The latter is concerned with hiding kilohertz signals in a megahertz environment, the former with hiding hertz signals in a kilohertz environment. It turns out that many spread spectrum techniques adapt quite well to data hiding in audio signals. In a normal communication channel, it is often desirable to concentrate the information in as narrow a region of the frequency spectrum as possible. The basic spread spectrum technique, on the other hand, is designed to encrypt a stream of information by spreading the encrypted data across as much of the frequency spectrum as possible.

While there are many different variations on the idea of spread spectrum communication, the one we concentrated on is Direct Sequence (DS). The DS method spreads the signal by multiplying it by a “chip,” a pseudo-random sequence modulated at a known rate. Since the host signals are in discrete-time format, we can use the sampling rate as the “temporal quanta” for coding. The result is that the most difficult problem in DS receiving, that of establishing the correct start and end of the chip quanta for phase locking purposes, is taken care of by the nature of the signal. Consequently, a much higher chip-rate, and an associated higher data rate, is possible.

3.4.1 Procedure

In DS, a “key” is needed to encode the information and the same “key” is needed to decode it. The key is pseudo-random noise that ideally has flat frequency response over the frequency range, i.e., white noise. The key is applied to the coded information to modulate the sequence into a spread spectrum sequence.

The DS method is as follows: First, the data to be embedded is coded as a binary string using error-correction coding so that errors caused by channel noise and host signal modification can be suppressed. Then, the code is multiplied by the carrier wave and the pseudo-random noise sequence, which has a wide frequency spectrum. As a consequence, the frequency spectrum of the data is spread over the available frequency band. Then, the spread data sequence is attenuated and added to the original file as additive random noise (See Figure 2). DS employs bi-phase shift keying since the phase of the signal alternates each time the modulated code alternates. For decoding, phase values ϕ_0 and $\phi_0+\pi$ are interpreted as a “0” or an “1” which is a coded binary string.

In the decoding stage, the following is assumed: (1) The pseudo random key has maximum randomness and flat frequency spectrum; (2) The key stream for the encoding is known by the receiver. Signal synchronization is done, and the start/stop point of the spread data are known; (3) The following parameters are known by the receiver: chip rate, data rate, carrier frequency, and the data interval.

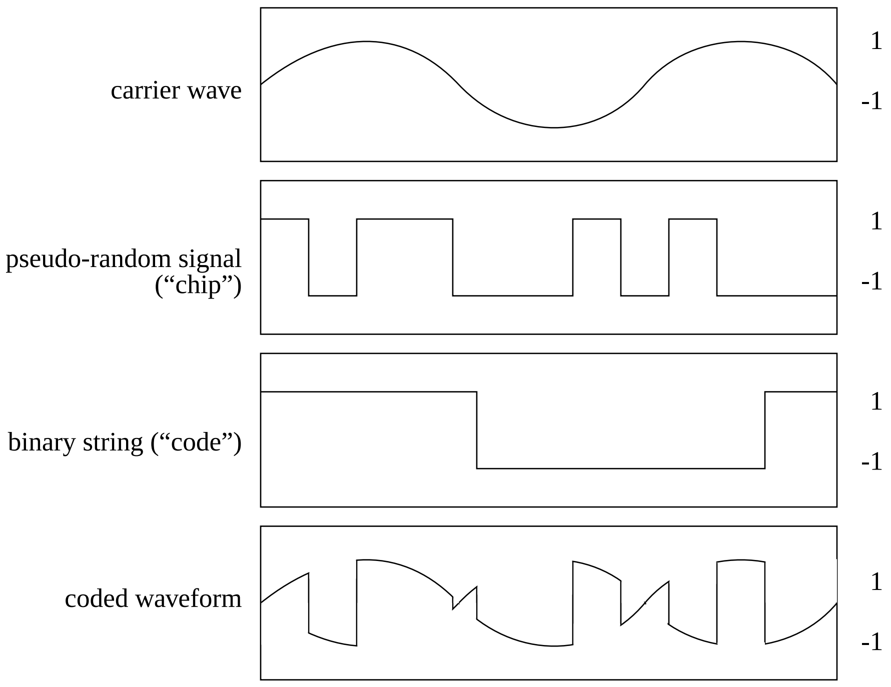

Figure 2: Synthesized spread spectrum information encoded by the Direct Sequence method.

3.4.2 Adaptive data attenuation

As mentioned above, the optimum attenuation factor varies as the noise level of the host sound changes. By adapting the attenuation to the short-term changes of the sound/noise level, we can keep the coded noise extremely low during the silent segments and increase the coded noise during the noisy segments. In our experiments, the quantized magnitude envelop of the host sound wave is used as a reference value for the adaptive attenuation; and the maximum noise level is set to 2% of the dynamic range of the host signal.

3.4.4 Redundancy and error correction coding

Unlike phase coding, DS introduces additive random noise to the sound. To keep the noise level low and inaudible, the spread code is attenuated (without adaptation) to roughly 0.5% of the dynamic range of the host sound file. The combination of simple repetition technique and error correction coding[1] ensure the integrity of the code. A short segment of the binary code string is concatenated and added to the host signal so that transient noise can be reduced by averaging over the segment in the decoding stage. The resulting data rate of the DS experiments is 4bps.

3.5 Sound context analysis

The detectability of noise inserted into a host audio signal is linearly dependent upon the original noise level of the host signal. To maximize the quantity of embedded data, while ensuring it is unnoticed, it is useful to express the noise level quantitatively. The noise level is characterized by computing the magnitude of change in adjacent samples of the host signal:

$$\sigma^2_{local} = \frac{1}{|Smax|} \times \frac{1}{N} \times \sum_{n=1}^{N-1} [s(n+1) - s(n)]^2$$

where N is the number of sample points in the sequence and S_{max} is the maximum magnitude in the sequence. We use this measure to categorize host audio signals by noise level (See Table 2).

Table 2: Audio noise level analysis

	studio quality		crowd noise
σ^2_{local}	<0.005	$0.005<\sigma^2_{local}<0.01$	0.01<

4. CONCLUSION

Several techniques are discussed as possible methods for embedding data in host image and audio signals. While we have had some degree of success, all of the proposed methods have limitations. The goal of achieving protection against intentional removal may be unobtainable.

Automatic detection of geometric and non-geometric modifications applied to the host signal after data hiding is a key data hiding technology. The optimum trade-offs between bit-rate, robustness, and perceivably need to be defined experimentally. The interaction between various data hiding technologies needs to be better understood.

While compression of image and audio content continues to reduce the necessary bandwidth associated with image and audio content, the need for a better contextual description of that content is increasing. Despite its current short-comings, data hiding technology is important as a carrier of these descriptions.

5. ACKNOWLEDGMENT

This work was supported in part by the MIT Media Laboratory’s News in the Future research consortium and IBM.

6. REFERENCES

1. P. Sweeney, Error Control Coding (An Introduction), Prentice-Hall International Ltd, 1991.
2. E. Adelson, *Digital signal encoding and decoding apparatus*, U. S. Patent 4,939,515, 1990.
3. FTP://sumex-aim.stanford.edu/stego.
4. W. Bender, *Data Hiding*, News in the Future, MIT Media Laboratory, (unpublished lecture notes), May, 1994.
5. A. Lippman, *Receiver-compatible enhanced definition television system*, U. S. Patent 5,010,405, 1991.
6. J. Gruber, *Smart Paper*, Wired, 2:12, December, 1994.
7. K. Matsui and K. Tanaka, *Video-Steganography: How to Secretly Embed a Signature in a Picture*, IMA Intellectual Property Project Proceedings, 1:1, January, 1994.
8. R. C. Dixon, Spread Spectrum Systems, John Wiley & Sons, Inc., 1976.
9. S. K. Marvin, Spread Spectrum Handbook, McGraw-Hill, Inc., 1985.
10. A. V. Oppenheim and R. W. Schafer, Digital Processing of Speech Signals, Prentice-Hall, Inc., 1975.