works best on sounds whose spectral components are close to those of the masking sound, but also occurs for components further away. The effect decreases more quickly toward lower frequencies than toward higher ones. The same is true for the time behavior: the masking is greatest for sounds which occur simultaneously, but can also be perceived in the time intervals shortly before and after the masking sound is supplied.

As stated, it can be deduced from the masking effect that there are signals which can be added inaudibly to an audio signal. The momentary power spectrum of these signals should therefore remain at all times under the masked threshold of that point in time. This means therefore that a data flow (series of bits) can also be added, that is, by constructing a signal of this kind from these bits. This can be done in the following way (see Fig. 1).

In order to use the masking effect, the signal is first split into subbands by means of filtering. The samples in each subband are then grouped into consecutive time windows (of approximately 10 ms in length). The windows from all subbands which represent the same time interval form blocks. For each block the power spectrum is calculated, which is then used to determine the masked threshold in each subband [6]. From this the maximum permitted power of a signal to be added can be obtained per subband, so that this can be constructed from the data flow. After the addition the subband signals are joined together again by a reconstruction filter bank to form a wide-band signal. On the premise that the implemented scheme determines the masked threshold correctly, the resulting wide-band signal will sound the same as the original audio signal. In the paper it is assumed that the used masking model is correct. Extensive listening tests, however, have confirmed this [7].

The signal to be added from the data flow and the set masked threshold is constructed as follows. A certain number of consecutive bits from the data flow are taken together to form words. Each word is interpreted as an address which indicates a unique sample value, as shown in Fig. 2. The series of bits is therefore converted into a series of samples via this word series. These data samples are then grouped into windows and added to the corresponding samples in the subband window of the original audio signal.

The number of bits n_b which are used to form one word depends on the set masked threshold in the subband and the difference Δ_b between the consecutive sample values (see Fig. 2; Δ_b will be indicated in the following by the bit step size). By assuming that the incoming series of bits has a uniform probability density distribution, a power

$$P_b = (2^{2n_b} - 1)\,\frac{\Delta_b^2}{12} \qquad (1)$$

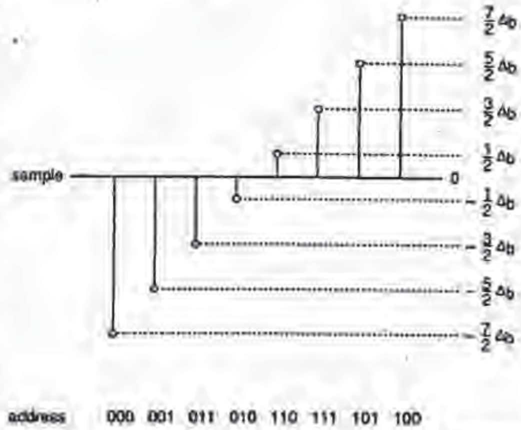


Fig. 2. Example as illustration of data sample construction with 3-bit words.

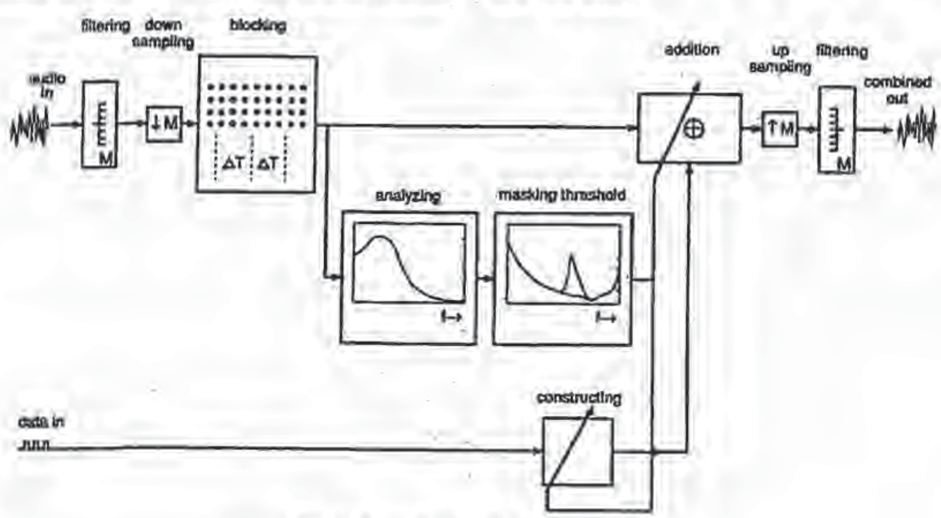


Fig. 1. Basic diagram for data addition.

can be assigned to each window of samples constructed in this way. With a given bit step size Δ_b, the number of bits per word is thus obtained from Eq. (1) as the maximum number n_b that still supplies a power under the set masked threshold in the corresponding subband. How the size of Δ_b is determined is discussed in Sec. 1.2.

The signal constructed in this way will have a power spectrum, the height of which is given by Eq. (1), but which is extended over the whole frequency range. However, the addition of the data signal at subband level limits the width of this spectrum to that of the subband. The grouping in subband time blocks is thus used not only to determine the masking properties of the audio signal, but also to modify the frequency–time characteristic of the data signals to be added.

The schematic diagram for retrieving the data added from the audio signal produced is shown in Fig. 3. The audio signal is first filtered in subbands and grouped in time windows, so that the same blocks are formed again (the filter banks to be used are of the (nearly) perfect reconstruction type [8]). After the position of the masked threshold has been determined, the sample values are extracted from the data signal as they were constructed during the addition. From the position of the masked threshold, the number of bits n_b that was added is again determined using Eq. (1). Finally, by using the same addressing table as that used during the addition (Fig. 2), the conversion to bit words can be made which, by placing them one after the other, again from the original data flow. Retrieval is thus obtained.

In order to distinguish between the added data sample value and the original audio sample value, it is necessary to apply a reference level in the combined signal. A level of this kind can be achieved by first quantizing the audio samples before carrying out the addition. In this case, quantizing can be described as

$$Q(s) = \Delta_Q \cdot \mathrm{ROUND}(s/\Delta_Q) \ , \tag{2}$$

where s is the value of the sample to be quantized, $Q(s)$ its value after quantization and Δ_Q the quantization step size. In order to distinguish between audio sample value and data sample value, a step size Δ_Q should be used which is greater than the range of possible data sample values:

$$\Delta_Q > 2\,\frac{2^{n_b}-1}{2}\,\Delta_b \ . \tag{3}$$

The data sample value can then be recognized as the "quantization noise," which results from quantizing the combined sample again (see Fig. 4).

The quantization of the audio signal reduces the accuracy of its representation, and this can be modeled as an increase in its noise level. Because the quantization has been used on a time-limited subband signal, this noise is however masked as long as its power remains under the masked threshold. (This property is also used with bit-rate reduction techniques [6].) The noise power is given as [9]

$$P_Q = \frac{\Delta_Q^2}{12} \ . \tag{4}$$

Because the quantization noise and the data signal are not correlated, the total power to be masked is obtained from the sum of their respective powers, given by Eqs. (1) and (4). Using Eq. (3), this power can be written as

$$P_t = P_b + P_Q < \frac{\Delta_Q^2}{6} \ . \tag{5}$$

The addition and retrieval parameters Δ_Q and n_b can therefore be determined as follows. After determining the masked threshold, the maximum possible quantization step size Δ_Q is determined using Eq. (5). The maximum number of n_b bits which can be added is

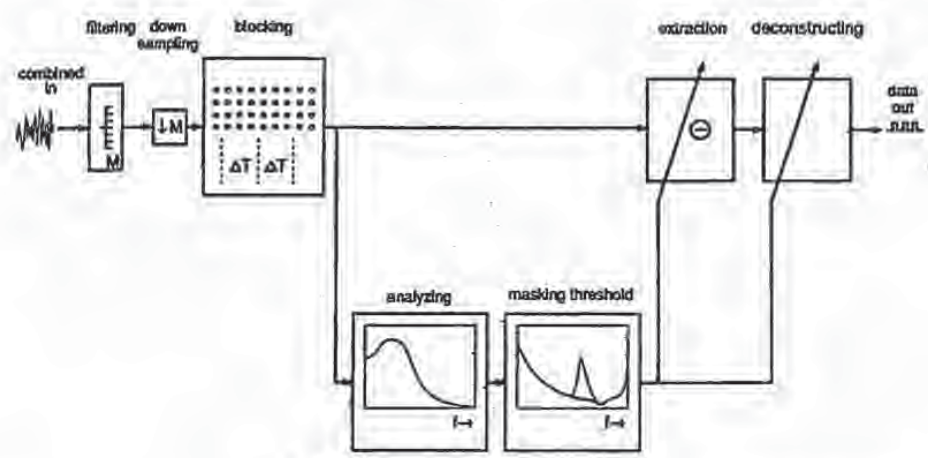


Fig. 3. Basic diagram for data retrieval.

then obtained from Eq. (3).

The resulting addition process can also be viewed as follows. It is determined for each sample value which part of its representation is significant and which part is not. This distinction is made possible by the masking effect: only a limited accuracy can be detected by the human ear. The insignificant part of the signal is then replaced by a different value, which indicates the information to be added.

1.2 Noise

The starting point is that the processing takes place with digital audio signals. This means that the combined signal produced will be quantized after the final filtering to a wide-band signal (see Fig. 1) to the representation accuracy of the transmission channel over which it will be sent. This creates quantization noise with, in the case of a channel with a linear quantization (PCM), a flat spectrum (that is, over the whole audio band) and a power P_N of [9]

$$P_N = \frac{\Delta_{ch}^2}{12} \tag{6}$$

in which Δ_{ch} indicates the quantization step size of the transmission channel.

The audio signal is filtered again in subbands at the receiver end (see Fig. 3) This affects the channel quantization noise in two ways. First, the probability density distribution of the noise will change into a Gaussian one and second, the power in each subband will decrease in proportion to the bandwidth of this subband. Thus in the case of a perfect transmission channel and a filtering in M subbands of equal width, the subband samples received have a noise component with a probability density function

$$p(\varepsilon) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-\varepsilon^2}{2\sigma^2}\right) , \tag{7a}$$

where ε is the magnitude and σ the standard deviation. σ is given by

$$\sigma = \sqrt{\frac{\Delta_{ch}^2}{12M}} . \tag{7b}$$

It is this standard deviation σ which determines the selection of the bit step size Δ_b in Eq. (1).

The data bits are recovered by converting the data samples received back to their address bit words according to a procedure as shown in Fig. 2. As a result of the noise, faults may occur in this process. By the use of a Gray code conversion [9] (Fig. 2) only 1 bit will toggle in the bit word each time the noise exceeds a decision threshold. (These thresholds lie in the middle between the noise-free sample values.)

Using Eq. (7a) an estimate can now be made of the error probability that n bits will be converted incorrectly ($n \geq 1$):

$$P(n) = \int_{-\infty}^{-(n-1/2)\Delta_b} p(\varepsilon)\, d\varepsilon + \int_{(n-1/2)\Delta_b}^{\infty} p(\varepsilon)\, d\varepsilon = 1 - \operatorname{erf}\left(\frac{2n-1}{2\sqrt{2}}\frac{\Delta_b}{\sigma}\right) . \tag{8}$$

Thus with σ according to Eq. (7b), Δ_b can be set for a certain error probability $P(n)$. On the other hand, Δ_b affects the number of bits n_b that can be added [see Eq. (3)]. As a result there is a tradeoff between n_b and $P(n)$.

In fact, the audio signal itself can be regarded as a "channel" over which the data are transported. A channel capacity C can then be defined as

$$C = \frac{1}{M} \sum_{m=0}^{M-1} n_{b,m} , \tag{9}$$

where M is the number of subbands and $n_{b,m}$ is the number of added bits per sample in subband m. According to Eq. (3), $n_{b,m}$ follows as

$$n_{b,m} = \text{TRUNC}\left[{}^2\log\left(\frac{\Delta_{Q,m}}{\Delta_{b,m}} + 1\right)\right] , \tag{10}$$

in which $\Delta_{Q,m}$ and $\Delta_{b,m}$ are the quantization step size and bit step size in subband m, respectively. If the subbands are all of equal width, then the channel noise σ [Eq. (7b)] is of equal strength in each subband and $\Delta_{b,m}$ can thus be taken the same in each subband [Eq.

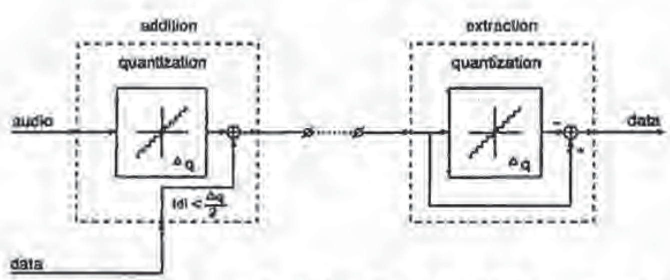


Fig. 4. Addition and extraction blocks from Figs. 1 and 3 in greater details.

(8)], $\Delta_{b,m} = \Delta_b = C_b\sigma$. Eq. (9) can now be written as

$$C = \frac{1}{M}\sum_{m=0}^{M-1} \mathrm{TRUNC}\left[{}^2\log\left(\frac{\sqrt{12M}}{C_b}\ \frac{\Delta_{Q,m}}{\Delta_{ch}} + 1\right)\right]$$

$$\approx {}^2\log\frac{\sqrt{12M}}{C_b} + \frac{1}{M}\sum_{m=0}^{M-1} {}^2\log\Delta_{Q,m} - {}^2\log\Delta_{ch}\ . \tag{11}$$

The first term reflects the effect of the channel noise. An increase in the parameter M, that is, splitting up the signal into more subbands, reduces the noise contribution in each band, which means that more bits can be added. This increases the complexity of the system and also the delay of the audio signal as a result of the narrow-band filtering. The coefficient C_b takes into account the tradeoff between the number of bits added and the error probability occurring. The second term indicates the masking effect of the audio signal: the greater the masking, the greater Δ_Q, and thus the more information can be added. (As a result of the filtering, addition is also possible if some bands have $\Delta_{Q,m} = 0$.) The third term indicates that an increase in the representation accuracy of the audio signal increases the channel capacity by approximately the same size. For example, representation with 18 bits instead of 16 (linear PCM) means a four-times reduction of Δ_{ch} and thus an increase of C by 2 bits. (It is assumed here that addition has already taken place in each subband.) In the case of a transmission channel in which the representation accuracy varies, such as, for example, in NICAM [10], it may be useful to normalize $\Delta_{Q,m}$ by Δ_{ch} to a new parameter, which means that the varying property can then be eliminated.

As stated, (nearly) perfect reconstruction filter banks are used [8]. This is necessary to ensure that the (subband) sample values used in the retrieval are (almost) the same as those which occurred after the addition (except for the wide-band quantization noise). In the filter structures used up and down sampling takes place (Figs. 1 and 3). This makes the system a multirate system. For a proper functioning the total delay between the two filters on both sides of the transmission channel must be a complete number of times the highest downsampling factor (M). In that case the delay at subband level is also a complete number of sample periods. Consequently, synchronization is required at the receiver end (processing in windows also makes this necessary). By not up and down sampling, this synchronization seems to be no longer required. However, the perfect reconstruction property will then be lost. Because of the processing (quantizing and adding) the spectrum of the subband samples changes over the whole bandwidth (given by the sampling frequency), while their filters only allow through the part in the corresponding subband. These two only coincide when sampling at the critical rate, and only then is perfect reconstruction possible. (The filter sequence for which the (nearly) perfect reconstruction property must apply is synthesis analysis, that is, the reverse to what the filter banks were designed for [8]. The fact that in this case the perfect reconstruction property is also valid can be seen by looking at the analysis–synthesis–analysis cascade. The first two filters form a perfect reconstruction pair as they were designed. The signals at the input of both analysis filters are therefore identical. Because the analysis filters are the same, it follows that the synthesis–analysis pair must also be a perfect reconstruction pair.)

A different approach to the one stated here is Nyquist's first criterion. From this it also follows that with an ideal bandpass filter no intersymbol interference occurs if the symbols are on (a multiple of) the critical rate (and are detected synchronously).

2 COMPATIBLE CODING

2.1 The Principle

Using the technique presented, a surround–stereo–surround coding system can now be developed which is very suitable for use in HDTV. Multichannel audio can be sent over a stereo transmission channel so that stereo reception is possible without additional modification, while there is the possibility of surround reception with a receiver equipped with additional electronics. In the following it will be assumed that the HDTV audio consists of five audio channels.

Fig. 5 shows the principle of the system. The programs are supplied with five-channel sound. A down mix to two-channel stereo is then made from this version. There are no restrictions on the way in which this down mix is made, that is, a signal with an optimum stereo effect can be produced. In addition to the stereo signal, a three-channel (audio) signal is also generated which, together with the stereo signal, contains all the information on the original five-channel composition. These information signals are then added to the stereo signal according to the technique described in Sec. 1 and retrieved at the receiver end.

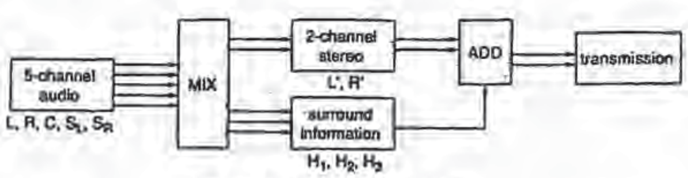


Fig. 5. Proposed coding scheme.

Because of the identical format, the signal transmitted is compatible and existing receivers can still be used. Reproduction of this signal will give the listener the stereo sensation as it was optimized during the down mix. Of course, the extra information is also reproduced but, because of the masking effect, the listener is not aware of this. This information is however still available by means of the technique described. The receiver must be expanded for this with additional electronics. After retrieving this information, the down mix carried out can be reversed, which means that the reproduction of the five-channel surround-sound sensation becomes possible.

2.2 The System

The original five audio channels are indicated with L, R, C, S_L, and S_R. Of these the first two signals are thought to be supplied to loudspeakers which are on the left and right of the video screen, respectively, the third (central) signal to a loudspeaker near the screen, and the latter two signals (surround) to the loudspeakers behind the listener (see Fig. 6). A stereo down mix could be

$$L' := L + \frac{1}{2}\sqrt{2}\,C + S_L \tag{12a}$$

$$R' := R + \frac{1}{2}\sqrt{2}\,C + S_R \,. \tag{12b}$$

(Other possibilities are conceivable.) Numerous signals can store the surround information here, but one possibility is

$$H_1 := C \tag{12c}$$

$$H_2 := S_L \tag{12d}$$

$$H_3 := S_R \,. \tag{12e}$$

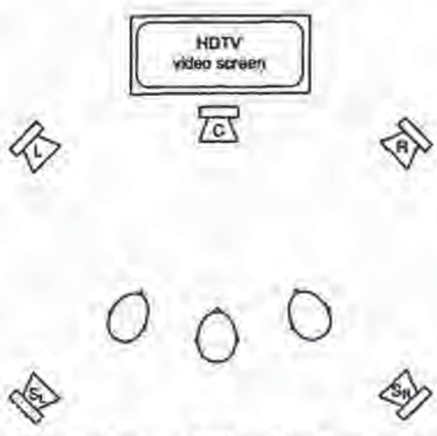


Fig. 6. Loudspeaker setup for five-channel surround sound.

In this case it is, of course, sensible to use first data reduction on C, S_L, and S_R [6]. The L' and R' signals are processed according to the method described in Sec. 1, and the information H_1, H_2, H_3 is added. After retrieving this information, the down mix can be reversed and the five-channel sensation can be produced again:

$$L'' := L' - \left(\frac{1}{2}\sqrt{2}\,H_1 + H_2\right) \tag{13a}$$

$$R'' := R' - \left(\frac{1}{2}\sqrt{2}\,H_1 + H_3\right) \tag{13b}$$

$$C'' := H_1 \tag{13c}$$

$$S''_L := H_2 \tag{13d}$$

$$S''_R := H_3 \,. \tag{13e}$$

A problem may occur as a result of this dematrixing. During the addition of the information, a quantization must be carried out (see Sec. 1.1). This quantization is carried out on the subband samples of L' and R' and in such a way that the resulting quantization noise is masked by these audio signals and thus remains inaudible. The stereo signal including the added information thus still creates the same listening experience. Dematrixing [Eqs. (13)] can however separate the audio signal from the quantization noise, which means that the noise could become audible. The effect becomes clear by looking at a silent channel (and switching off the other loudspeakers when listening). Assume, for example, that all channels with the exception of channel C are silent. In that case L' and R' are both equal to $\frac{1}{2}\sqrt{2}C$ [see Eqs. (12a,b)]. These signals are quantized and H_1 ($= C$), H_2 (silent) and H_3 (silent) are added. After retrieval, C, S_L, and S_R are determined from H_1, H_2, and H_3. The result is used to reverse the down-mixing. This dematrixing will remove $(\frac{1}{2}\sqrt{2}H_1 + H_{2,3}) = \frac{1}{2}\sqrt{2}C$ from L' and R' [see Eqs. (13a,b)]. As a result of this the quantization noise produced during the addition procedure remains in the left and right channel L'' and R'', while the signal that masked this noise, $\frac{1}{2}\sqrt{2}C$, is now transmitted to another loudspeaker, C''. Because the audio signal is still present, it will still have a masking effect on the quantization noise, though this will be less effective than if they were both generated by the same loudspeaker.

A remedy is to expand the information signals $H_{1,2,3}$ with some extra control information. This information then indicates which channels are silent, so that after dematrixing, any residual sound can be removed from these channels. Possibly the information is given for every subband separately. In addition, instead of always coding C, S_L, and S_R in H_1, H_2, and H_3, it is better to take the weakest three of L, R, C, S_L, and S_R. This ensures that the quantization noise is always in those signals which give the greatest masking and therefore that the chance of its audibility is limited. The choice made is added as control information to $H_{1,2,3}$ and used during dematrixing. Informal listening tests on various types of program material have proven the validity of this procedure. Only by switching off some channels, it could occur that noises in the other channels became audible. Those cases only happened with especially constructed signals. Common audio signals did not reveal any problem.

A complete abundance of audible quantization noise is possible by adapting the (audio) input of the masking model [11]. Instead of the power spectrum of the down-mixed stereo signal, that of the signal which will remain after dematrixing should be used. For example, in the case described by Eqs. (12) and (13) the power spectrum of L and R instead of L' and R' should be taken to determine the masked threshold.

A final question is whether there is always sufficient room available in the stereo signal to add the information. As explained in Sec. 1 with Eq. (11), this amount of room depends on two main factors, namely, the masking power of the audio signal and its representation accuracy [Δ_Q and Δ_{ch} in Eq. (11)]. It is clear that a higher representation accuracy simplifies the task because the amount of information to be added is independent of it. Experiments have, however, shown that the representations currently used offer sufficient space for the information required. With regard to the masking power of the audio signal, one might naively expect there to be problems with low masking power. In this application, however, the information to be added, H_1, H_2, and H_3, is an audio signal which is also present in the masking signal itself, L' and R'. In other words, if there is limited masking, that is, if little room is available, there is also little information to be added. In the extreme case of no masking (L, R, C, S_L, and S_R are all silent), for example, there is also no need to add information. Another example is given by assuming L and R to contain the direct sound and early reflections and S_L and S_R to contain the reverberation of a concert-hall recording. When the music stops, there is still a (decreasing) reverberation. However, in the down-mixed stereo signal L' and R', this reverberation is also present and as a result there is still an audio signal in order to mask the information to be added (which information is that L and R are silent!).

Within the European HDTV project EUREKA-95, the system is considered as a potential way to transmit HDTV sound. Its interesting feature is the compatibility to the two-channel D2MAC transmission standard. After various informal listening tests, which showed the system's potential, a formal listening test on the system's performance was organized by EU95. During the summer of 1990 these tests have been conducted. Critical signals were constructed. The tests did not reveal any significant audible degradation of these signals after having been mixed into a two-channel NICAM stereo signal. Further formal listening tests are planned for early 1992.

3 CONCLUSIONS

A new surround–stereo–surround coding technique is presented. The down mix to the stereo signal may be optimized to give the best stereo effect. The extra information required to reproduce the original multichannel surround sensation using the stereo signal is added in this stereo signal. Here the masking effect is used so that the addition remains inaudible. Compatibility with current stereo standards is therefore guaranteed. Using the system it is possible to maintain the original channel separation.

4 ACKNOWLEDGMENT

The authors would like to express their thanks to Dr. W. F. Druyvesteyn, who came up with the idea of using the masking effect for information addition, and to Dr. R. N. J. Veldhuis, who devised the basic algorithms for this addition.

5 REFERENCES

[1] E. Stetter, "Mehrkanal-Stereoton zum Bild für Kino und Fernsehen" (Multichannel Stereo Sound for Cinema and Television Picture)," *Rundfunktech. Mitt.*, vol. 35, pp. 1–9 (1991).

[2] D. J. Meares, "Sound Systems for High Definition Television," *Acoust. Bull.*, vol. 15, pp. 6–11 (1990).

[3] W. R. Th. ten Kate, L. M. van de Kerkhof, and F. F. M. Zijderveld, "Digital Audio Carrying Extra Information," in *Proc. ICASSP90* (Albuquerque, NM, 1990 Apr.), pp. 1097–1100.

[4] B. C. J. Moore, *An Introduction to the Psychology of Hearing*, 3rd ed. (Academic Press, London, 1989).

[5] E. Zwicker and H. Fastl, *Psychoacoustics Facts and Models* (Springer, Berlin, 1990).

[6] R. N. J. Veldhuis, M. Breeuwer, and R. van der Waal, "Subband Coding of Digital Audio Signals," *Philips J. Res.*, vol. 44, pp. 329–343 (1989).

[7] C. Gerwin and T. Rydén, "Subjective Assessments on Low Bit-Rate Audio Codecs," in *Proc. 10th Int. AES Conf. on Images of Audio* (London, 1991 Sept.), pp. 91–102.

[8] M. Vetterli and D. LeGall, "Perfect Reconstruction FIR Filter Banks: Some Properties and Factorizations," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-37, pp. 1057–1071 (1989).

[9] N. S. Jayant and P. Noll, *Digital Coding of Waveforms.* (Prentice-Hall, Englewood Cliffs, NJ, 1984).

[10] C. R. Caine, A. R. English, and J. W. H. O'Clarey, "NICAM 3: Near-Instantaneously Companded Digital Transmission System for High-Quality Sound Programmes," *Rad. Elec. Eng.*, vol. 50, pp. 519–530 (1980).

[11] W. R. Th. ten Kate, P. M. Boers, A. Mäkivirta, J. Kuusama, E. Sørensen, and K. E. Christensen, "Matrixing of Bit Rate Reduced Audio Signals," in *Proc. ICASSP92* (San Francisco, CA, 1992 March).

THE AUTHORS

W. R. Th. ten Kate

L. M. van de Kerkhof

F. F. M. Zijderveld

Warner R. Th. ten Kate was born in Leiden, The Netherlands, in 1959. He studied electrical engineering at Delft University of Technology, graduating in 1982 cum laude, and received the 1983 prize awarded by the Delft University Fund. During the final stages of his studies his research was directed at solar cells of amorphous silicon and silicon radiation detectors. He received the Ph.D. degree in 1987.

Since 1988 Dr. ten Kate has been working in the Acoustics Group of Philips Research Laboratories. In 1985 he also began studying the French horn at the Royal Conservatory in The Hague and graduated in 1989 with distinction.

•

Leon M. van de Kerkhof was born in Eindhoven, The Netherlands, in 1958. In 1978 he joined Philips Research Laboratories, where he worked on noise control (including reactive sound absorbers and aerodynamic noise) and the use of adaptive filters in acoustics. At the same time he began an evening course in electrical engineering at the Institute of Technology. After graduating in 1981 he continued his studies at the Eindhoven University of Technology and received a degree in 1987 cum laude. He then moved to Philips Consumer Electronics. His activities are in the sphere of digital audio, in particular audio source coding and HDTV sound. He is involved in various international projects, including Eureka 95 (HDTV), Eureka 147 (Digital Audio Broadcasting), JESSI AE14 (JESSI DAB), and ISO/MPEG Audio.

•

Franc F. M. Zijderveld was born in Helmond, The Netherlands, on 1961 November 20. In 1985 he completed his studies in electrical engineering at the Eindhoven Institute of Technology, his final project being the realization of an autofocus system for a CCD video camera. He then joined Philips Consumer Electronics, where he worked in the development laboratory for video equipment and was mainly involved in analog video signal processing in CCD cameras. In 1987 he moved to the Audio Signal Processing Group at the Philips Consumer Electronics Advanced Development Centre, working on the installation of an experimental four-channel audio postproduction room and on the digital 4–2–4 system. His current interest is centered on digital audio broadcasting.

A High-Rate Buried Data Channel for Audio CD

Preprint 3551 (D3-1)

Michael A. Gerzon
Technical Consultant, Oxford, United Kingdom
Peter G. Craven
Oxon, United Kingdom

Presented at the 94th Convention 1993 March 16–19 Berlin

AES

AN AUDIO ENGINEERING SOCIETY PREPRINT

A High-Rate Buried Data Channel for Audio CD

Michael A. Gerzon
Technical Consultant, 57 Juxon St., Oxford OX2 6DJ, UK
Peter G. Craven
11 Wessex Way, Grove, Wantage, Oxon OX12 0BS, UK

Abstract

The paper describes a new proposal for burying a high data rate data channel (with up to 360 kbit/s or more) compatibly within the data stream of an audio CD without significant impairment of existing CD performance. The new data channel may be used for high-quality data-reduced related audio channels, or even for data-compressed video or computer data, while retaining compatibility with existing audio CD players. The theory of the new channel coding technique is described.

0. Introduction

The paper describes a new proposal for burying a high data rate data channel (with up to 360 kbit/s or more) compatibly within the data stream of an audio CD without significant impairment of existing CD performance. The proposal in this paper is to replace a number (up to four per channel) of the least significant bits (LSBs) of the audio words by other data, and to use the psychoacoustic noise shaping techniques associated with noise shaped subtractive dither to reduce the audibility of the resulting added noise down to a subjective perceived level equal to that of conventional CD.

Simply replacing the LSBs of existing audio data would, of course cause a drastic audible modification of the existing audio signal for two reasons :

1) the wordlength of existing signals would be truncated to (say) only 12 bits, which would not only reduce the basic quantization resolution by 24 dB, but also would introduce the problems of added distortion and modulation noise caused by truncation (e.g. see refs. [1-4]).

2) Additionally, the replaced last (say) 4 LSBs would themselves constitute an added noise signal, which itself may not have a perceptually desirable random-noise like quality, and will also add to the perceived noise level in the main audio signal, typically increasing the noise by a further 3 dB above that due to truncation alone, giving in this case as much as 27 dB degradation total in noise performance.

This paper describes methods of overcoming all these problem in replacing the last few LSBs of an audio signal by other data. The new method involves the following steps:

A) Using a pseudo-random encode/decode process, operating only on the LSB data stream itself without extra synchronizing signals, to make the added LSB data effectively of random noise form, so that the added signal becomes truly noise-like.

B) Using this pseudo-random data signal as a subtractive dither signal (e.g. see [1-4]), so that simultaneously it does not add to the perceived noise and that it removes all nonlinear distortion and modulation noise effects caused by truncation. Remarkably, and unlike in the ordinary subtractive dither case [3], this does not require the use of a special subtractive dither decoder, so that the process works on a standard off-the-shelf CD player, and

C) additionally, at the encoding stage, incorporating psychoacoustically optimized noise shaping of the (subtractive) truncation error, thereby reducing the perceived truncation noise error by around 17 dB further.

The overall effect of combining these three processes is that if one incorporates data into the last few LSBs, then the effects of distortion, modulation noise and perceived audible patterns in the LSB data are completely removed, and the resulting perceived steady noise is reduced by around 23 dB below that of ordinary unshaped optimally dithered quantization to the same number of bits. For example, when the last 4 LSBs of the 16 bit CD wordlength is used for buried-channel data, the perceived S/N (signal-to-noise ratio) is around 91 dB - approximately the same as ordinary 16 bit CD quality when unshaped dither is used.

The result of this process is that as much as 2 x 4 = 8 bits of data per stereo sample is available for buried data without significant loss of audio quality on CD, giving a data rate of 8 x 44.1 = 352.8 kbit/s.

While the new process achieves potentially high data rates for the buried channel, it does of course reduce room for improvements in CD audio quality approaching 20 bits effective audio quality, such as described in refs. [3],[4]. However, there is no reason why the process should only be used with one fixed number of LSBs, and by reducing the data rate of the buried channel to a smaller number of LSBs, one correspondingly improves the resolution of the audio - for example achieving an effective perceived S/N of around 103 dB for a system using 2 LSBs of data per signal channel sample, with a data rate still of 181.4 kbit/s.

One can even make the number of LSBs used fractional, say, ¼ or ½ or 1½ LSBs per sample. This may be used either to precisely match the buried channel to a desired data rate, or to minimize the loss of audio quality, especially at very low data rates.

Additionally, by including in the LSB data channel itself low-rate data indicating the number of LSBs "stolen" from the main audio channels, it is possible to vary the number of LSBs stolen in a time-variant way, so that, for example, more LSBs can be taken by the buried channel when the resulting error is masked by a high-level main audio signal. The noise-shaping can also be varied adaptively at the encoding stage so that at high audio levels, the noise error is maximally masked by the audio signal, thereby increasing the data rate of the buried channel during loud passages to, in some cases, as much as 720 kbit/s.

It is also shown in this paper that with stereo signals, it is possible to code data jointly in the least significant parts of the audio words of the two (or more) channels, using a multichannel version of the data encoding process involving the use of vector quantizers and subtractive vector dithering by a multichannel pseudo random data signal for the dithering. The basic theory of vector dithering is described in section 5, although readers may find it best to omit these technically difficult aspects on first reading. It is shown that the vector multichannel version of the data coding process ensures left/right symmetry of any added noise in the audio reproduction, and an advantageous noise performance.

The approach described in this paper is substantially different from an alternative method of burying data described in [5], which involved a process of splitting the audio signal into subbands, replacing the LSBs of the subbands with data based on auditory masking theory, and then reassembling the resulting data by recombining the subbands. Not only is that process very complicated, with a considerable time-delay penalty in the subband encoding/decoding process, but it has to be done with extraordinary precision to prevent data errors in the band splitting and recombining process. By contrast, the present process involves little time delay, involves relatively simple signal processing, and further is such as to guarantee the lack of audible side-effects due to nonlinear distortion, modulation noise or data-related audible patterns.

Another approach to transmitting data in an audio waveform, for use with the NICAM system, has been described by Emmett [24], in which the shape of the error spectrum is adaptively changed to be masked by the audio signal.. This may or may not have some common features with the present proposal, although the details of Emmett's proposal are not clear from his published preprint. However, according to Emmett [24], to attain a high encoded data rate with his proposal requires using a data rate that changes with signal level so as not to be audible during quiet passages. The present proposal does not require the use of such level-adaptive data rates.

1. Uses Of Buried Data

1.1 Advantages over CD ROM media

The availability of a buried data channel with data rates of the order of 360 kbit/s without significant loss of audio quality on audio CDs, fully compatible with conventional playback on standard audio players, opens up prospects for many new products. Unlike standards such as CDI based on CD-ROM, the additional data can be added without destroying compatibility with playback over tens of millions of existing audio players. This means that the new data channel can be added while still giving the CD the advantages of mass-market economies of scale of production, thanks to the existing audio-only market. Thus applications using the new data channel should result in much lower prices than for media where the number of players is limited.

1.2 Application to multichannel sound

One application of the new data channel is using the additional bits to add, using audio data compression, additional audio channels for three- or more-speaker frontal stereo or surround sound, such as described for example in [6],[7],[8]. Because CD has higher quality than available data compression systems (despite spurious claims of "transparency" or "CD quality" by some less cautious proponents of such systems), care must be taken that the additional channels are not too compromised in quality by the data compression process, which means that a rather lesser degree of compression is desirable than for DAB or film surround-sound. However, since two of the transmitted audio channels are the standard CD audio channels and the design of the buried channel avoids nonlinear or modulation noise effects on these main channels, all the data rate in the buried channel can be used solely for the additional channels, giving each a higher data rate than if the buried channel were used to transmit the whole audio signal. In using the buried channel to transmit additional directional audio channels, it is important to design the codec error signals so that they do not become audible through the mechanism of directional unmasking described in three of one of the author's references [9],[10],[11].

The data rate available is sufficient to transmit a Dolby AC-3 or MUSICAM surround 5-channel surround-sound signal, but these systems involve a quality compromise with the data rate, so that this is not a preferred procedure.

High-quality data compressed additional audio channels can, unlike existing data compression systems, minimize the risk of destruction of subtle auditory cues such as those for perceived distance (see [12]), thereby maintaining CD digital audio as the preferred medium for high quality audio, while adding additional channels. For high quality (and especially musical) use, it may be preferred to use additional buried audio channels either for frontal-stage 3- or 4- speaker stereo or for 3-channel horizontal or 4-channel full-sphere with height [13],[14] ambisonic surround sound (see refs. [7],[8],[15]), rather than for the rather cruder theatrical "surround-sound" effects considered appropriate for cinema or video-related surround-sound systems. However, systems have been proposed for intercompatible use of both kinds of system [7], [8].

Since the main audio channels in this proposal convey high-quality audio, it is possible to use the spectral envelope of the main audio channels to convey most or all of the dynamic ranging information used for the subbands in data reduction systems for related subsidiary channels conveyed in the buried data channel, especially if the main audio channels incorporate a mixture of all the transmitted channels so that no direction is canceled out. This saves the data overhead of conveying ranging data, which in high quality systems may save of the order of 60kbit/s, as compared to a stand-alone data compression system. This will allow a system conveying n related channels using 4 LSBs per main CD audio channel to give a performance equivalent to that of a stand-alone data compression system conveying n-2 channels in about 420 kbit/s. For 3-channel systems, such as horizontal B-format surround-sound or 3-channel UHJ [15] or frontal-stage 3-channel stereo, this quality is unlikely to be audibly distinguishable from an uncompressed data channel, and for 4-channel systems, the results will still subjectively approach that of critical studio-quality material, and even for 5-channel material, the results will be considerably less compromised than that for DAB or cinema surround-sound, using a data rate for the additional channels of well over twice that use in those applications.

1.3 Video and computer data

Alternatively, the buried data channel can be used for conveying related computer data, such as graphics, multilingual text or track copyright information. Because of the high available data rate, this can be done with very much higher quality than is possible on the subcode channels of CD, conveying for example with JPEG image data compression of the order of one high quality color photographic image per second. A data rate of 360 kbit/s is even enough to convey a reasonable video image. Using the existing MPEG standard, this would have very low resolution (although certainly good enough for moving inserts within a still image), but near-future image data-compression methods based on using the highly non-Gaussian nature of images are expected to make consumer-quality video available within this data rate.

1.4 Dynamic range data

Another use would be to convey dynamic-range reduction or enhancement data, e.g. a channel conveying the setting of a gain moment by moment. This would allow the same CD automatically to be played with different degrees of dynamic compression according to environment, by choosing the gain adjustment channel appropriate to that environment. This would include the possibility of completely uncompressed quality for high-quality use, without making the CD incompatible for more normal use, e.g. in broadcasting. An advantage of providing the dynamic range gain data in the data subchannel rather than using automated dynamic range modification algorithms is that one can always do a much better subjective job using manual intervention based on a knowledge of the music and its needs, but at the expense only of considerable time and effort. This effort can be recorded for consumer use in the buried data channel. If automated algorithms are used for the dynamic range gain conveyed by the buried data channel, these can be of a much more sophisticated and subtle nature than those normally available to the consumer (e.g. [16]).

1.5 Frequency Range Extension

A further use related to the original audio would be to add in the subchannel data-reduced information allowing information above 20 kHz to be reconstructed. One of the limitations of compact disc is that the frequency range is limited to 20 kHz. Although the ears' sine wave hearing is, for all except a small minority of (generally young and often female and/or asthmatic) listeners, limited to below 20 kHz, this does not mean that there is no loss of perceived quality caused by the sharp bandlimiting to 20 kHz. It is widely noted that there is a significant loss of perceived quality when comparing high-quality digital signals sampled at say 44.1 kHz as compared to 88.2 kHz.

From a quality viewpoint, it may be more important to use an extended bandwidth to provide a more gentle roll-off rate than to provide a response flat to 40 kHz, since (unlike the brickwall filters used with ordinary CD), such gentler roll-offs are similar to those encountered in natural acoustical situations.

The extended bandwidth can be provided by using a high-order complementary mirror filter pair of the kind described in Regalia et al. [21] and in Crochiere and Rabiner [22] to split an 88.2 kHz-rate sampled digital signal into two bands sampled at 44.1 kHz. The filters involved will overlap, although using a high-order filter [21], the region of significant overlap can be reduced to of the order of a kHz. Within the overlap region there will be aliasing from the other frequency range, although the reconstruction of the full bandwidth [21,22] will cancel out this aliasing. The band below 22.05 kHz can then be transmitted as the conventional audio, and the band above 22.05 kHz can be transmitted in data reduced form in the buried data channel at a reduced data rate of, say, between 1 and 4 bits per sample per channel, using known sub-band or predictive coding methods. Phase compensation inverse to the phase response of the low pass filter in the complementary filter pair may be employed to linearize the phase response of the main sub-22.05 kHz signal for improved results for standard listeners, with the use of an inverse phase compensating filter in the decoding process for reconstructing the wider bandwidth signal.

1.6 Combined applications

Any or all of these uses can, of course, be combined, subject only to the restrictions of the data rate, so that the buried data channel could be used for example to convey one additional audio channel, a dynamic range gain signal, extended bandwidth and additional graphics, text (possibly in several languages), copyright and even insert video data as appropriate.

For historical material, where the dynamic range may be significantly less than 90 dB, it may even be possible to increase the data rate available further by allocating even more bits to the buried data channel, since an increased noise level may not be significant. For this reason, it may be desirable to allow the possibility of allocating as many as 12 or even 16 bits of audio data (say bits 10 to15 or even 8 to 15 of each audio channel) to the buried data channel.

2. Pseudo-Random Coding of Data

2.1 Pseudo-Randomized data

It is essential, if the LSBs of an audio signal are to be replaced by data, that the replacing data should truly resemble a random noise signal (albeit perhaps one that may be spectrally shaped for psychoacoustic reasons). Most data signals, when listened to as though they were digital audio signals, have some degree of systematic pattern which may well prevent them from sounding or behaving truly like random noise. Such departures from random noise like behavior are generally much more perceptually disturbing or distracting than a simple steady noise.

Also, if we can ensure that added data behaves like a noise signal with known statistical properties, one can use all that is known in the literature on dither and noise shaping (see [1]-[4],[17]-[20]) to optimize the perceptual properties of the added data to minimize its audible effects.

The data signal is rendered pseudo-random with predictable statistics in our proposal by a data encode/decode process, the encode process having the effect of pseudo-randomizing the data signal, and the decode process having the effect of recovering the original data signal from the pseudo-randomized data signal, as in figure 1. From a practical point of view, it is highly desirable that the encode and decode process require no use of an external synchronizing signal, but that the decode process should work entirely from the pseudo-randomized data sequence itself.

The simplest way of constructing such an encode/decode pseudo-randomizing process for data is to use a cyclic pseudo-random logic sequence generator separately on each bit. For example, if its input is zero, fig.2 shows a well-known binary pseudo-random logic sequence generator using feedback around three logic elements and a total shift register delay of 16 samples (a 1-sample delay is denoted by the usual notation z^{-1}). Provided that the logic state in the 16 samples stored in the shift register is not all zero, this binary sequence generator has the 16 logic states cycle through all $2^{16}-1$ = 65,535 non-zero states in a pseudo-random manner.

If, instead of using a zero input, the pseudo-random sequence generator of fig. 2 is fed with a binary data stream s_n, then it has the effect of a pseudo-randomizer for the input data. This encoding scheme is based on the recursive logic

$$t_n = s_n \oplus t_{n-1} \oplus t_{n-3} \oplus t_{n-14} \oplus t_{n-16}, \qquad (2\text{-}1)$$

where t_n is the output binary logic value of the network at integer sample time n, s_n is the input binary logic value of the network at integer sample time n, and $\oplus$ represents the logic "exclusive or" or Boolean addition operator (with truth table $0 \oplus 0 = 1 \oplus 1 = 0$, $0 \oplus 1 = 1 \oplus 0 = 1$).

Conversely, if exactly the same arrangement of logic gates is fed with the pseudo-randomized data t'_n, then the effect of the "exclusive or" gates on the input signal is to restore the original data stream. This is achieved by the inverse decoding logic process

$$s_n = t_n \oplus t_{n-1} \oplus t_{n-3} \oplus t_{n-14} \oplus t_{n-16}, \qquad (2\text{-}2)$$

illustrated in the second diagram in fig. 2.

Thus by using a logic network recursively with a total of L = 16 samples delay and only 4 "exclusive or" gates, a binary data stream can be pseudo-randomized, and the same network can decode the data stream back to its original form. For constant signals, there is a one in 65,536 chance that the undesirable non-random zero state will be encountered, but this low probability is probably acceptable, given that even a single binary digit change of input is likely to "jog" the system back into a pseudo-random output state.

Other well-known pseudo-random binary sequence logic generators with shift registers of longer length L than 16 samples can be used for encoding and decoding in the same way, with their fed-back output given by subjecting the delayed sequence output and the input to a "sum" logic gate. Such length L sequences will have, for a constant input, only one chance in 2^L-1 of giving an unrandomized output, and will have a sequence length of 2^L-1 samples.

Although the pseudorandom binary sequence generator described in (2-1) and fig. 2 is a maximum length sequence for a zero input, it has a shorter length for an all-one constant input, and in general, the precise behavior with, say periodic inputs is hard to predict. Partly for this reason, it is not absolutely essential to use a maximum-length sequence generator, provided that the length of the sequence is not too short for constant inputs

It will be noted that the network of fig. 2 only has L = 16 samples of memory, so that when used as a decoder, any data errors in the input will only propagate for L samples, and then the output will recover. This lack of long-term memory in the decoding process means that there are no special requirements on the error-rate of the transmission channel. Because of the small number of logic elements in fig. 2, a single sample error in the received data stream will only cause five sample errors in the decoded output.

As shown in fig. 3, typically, for use with CD, the data will first be arranged to form a number of bits of data per sample of each audio channel, for example 8 bits of data constituting bits 12 to 15 of the left and the right audio channels (where bit 0 is the most significant bit (MSB) of a 16 bit audio word and bit 15 the least significant bit).

Then each of these (say 8) bits will, separately, be encoded by a pseudo-random logic such as that of fig. 2 to form a pseudo random sequence, and the resulting pseudo-randomized bits used to replace the original bits in (say) bits 12 to 15 of the left and the right audio channels. The resulting noise signals in the left and right audio channels will be termed the (left and right) data noise signals.

Alternatively, instead of pseudo-randomizing individual bits of the audio words representing data separately, they can be pseudo-randomized jointly by regarding the successive data bits of a word as being ordered sequentially in time, and applying a pseudo-random encoder such as that of figure 2 to this sequence of bits. For example, eight bits of data per audio sample can be sequentially ordered before the next eight bits of data corresponding to the next audio sample, and the pseudo random-logic encoding can be applied to this time series of bits at eight times the audio sampling rate.

An advantage of this strategy is that errors in received audio samples propagate for (in this example) for only one eighth of the time as in the case where each word bit is separately pseudo-randomized.

M-level data signals, taking one of M possible values, conveying $\log_2 M$ bits per sample can also be pseudo-randomized by a direct process involving congruence techniques, whereby the coded version w'_n of the current sample M-level word w_n is given by

$$w'_n = w_n + \sum_{j=1}^{L} a_j w'_{n-j} \pmod{M}, \qquad (2\text{-}3)$$

where the a_j's are (modulo M) integer coefficients chosen (if necessary by empirical trial-and-error) to ensure that all M possible constant inputs result in a pseudo-randomized output with reasonably long sequence lengths. The inverse decoding of the pseudo-randomized M-level words is

$$w_n = w'_n - \sum_{j=1}^{L} a_j w'_{n-j} \pmod{M}. \qquad (2\text{-}4)$$

The logic techniques described with reference to figure 2 are just the special case when M = 2 of this more general congruence technique. The congruence technique can result in sequence lengths for constant inputs of length up to a maximum of M^L-1 samples, so that in general, the larger the value of M, the smaller need be L, with a consequent shortening of the time duration of propagation of errors.

A slightly more complex pseudo-randomization of data will provide an initial pseudo-randomization of M-level data by a method such one of those described here, and follow it by an additional one-to-one map between the M possible data values. The decoding will first subject the M levels to an inverse map before applying the inverse of the above pseudo-random encodings.

There are many similar but more complicated methods of pseudo-randomization of data streams, and as we have seen, these need have no coding delay or increase in data rate after coding, and can limit the duration of any errors in received data in the inversely decoded output to not more than a few samples after the occurrence of an erroneous audio sample.

As audio signals, the resulting pseudo-randomized data noise signals have a steady white noise spectrum and a (discrete) uniform or rectangular PDF (probability distribution function), in the example case described above having 16 levels in each of the left and right channels. Such discrete noise does not have the ideal properties of rectangular dither noise, although Wannamaker *et al* [17] have shown that it approximates many of these desirable properties in a precise mathematical sense. However, adding to it an extra random or pseudo-random white rectangular PDF noise signal with peak level ± ½ LSB converts it into noise with a true rectangular PDF with peak levels (in this example) of ± 8 LSB. In this case the added noise to convert from a discrete to a continuous PDF is at a very low level, being 24 dB below the level of the data noise signal.

2.2 Stereo parity coding

Although in the above example, we have described data being conveyed on each audio word bit of the data signal separately, it will be realized that data can alternatively be conveyed by more complicated combinations of the least significant digits (in any numerical base M, not just the binary base 2) of audio words, for example on the Boolean sum of the corresponding bit in the left and right audio signal.

For example, consider the case that a data rate of only one bit per stereo audio sample is required. Such a signal can be conveyed as the Boolean sum of the LSB in the left and the right audio channels, leaving the values of the LSB in individual audio channels separately unconstrained. Conveying a data channel using the Boolean sum of the corresponding bits of the left and right audio signals is herein termed stereo parity coding.

It is of course desirable that the effect on the conventional audio of reallocating bits to a buried data channel should be left/right symmetrical. In particular, if a buried data channel is used with a data rate of just one BPSS (bit per stereo sample), then one does not wish to code the data in the LSBs of only one of the two stereo channels. If the value of the respective N'th bits of the respective left and right channel signals are denoted by L^N_n and R^N_n at time n, then one codes a pseudo-randomized one bit per sample data channel t^N_n as

$$t^N_n = L^N_n \oplus R^N_n . \qquad (2\text{-}5)$$

If desired, an additional second pseudo-randomized one bit per sample data channel u^N_n can be encoded in the N'th bits of the stereo audio signal say as

$$u^N_n = L^N_n . \qquad (2\text{-}6)$$

in which case the data can be encoded via $L^N_n = u^N_n$ $R^N_n = L^N_n \oplus t^N_n$. and decoded via $u^N_n = L^N_n$. $t^N_n = L^N_n \oplus R^N_n$. Alternatively u^N_n can be encoded as R^N_n . The use of stereo parity encoding allows the separate one BPSS data channels to be separately decoded while maintaining left/right symmetry in the audio when an odd number of one BPSS channels are used.

One could standardize a basic one BPSS data channel as being conveyed via the parity (Boolean sum) of the LSBs (i.e. bit 15) of the left and right audio channels. Information about the way other data channels conveying more BPSS are coded will, in such a standardization, be conveyed by this basic data channel. By this means, a data decoder can read from the basic one BPSS stereo parity data channel how to decode any other data channels (if any) present. In particular, this allows if desired moment-by moment variation of the data rate, either adaptively to the amount of data needing transmission or adaptively to the audio signal according to its varying ability to mask the error signal caused by the hidden data channels.

For example, in loud passages in pop/rock music, the data rate allocated to say a video signal could be increased, allowing quite high quality video images in, say, heavy metal music.

2.3 Fractional bit rates

There is no reason why the buried data channels should be restricted to data rates of an integer number of BPSS, although this may be a convenient implementation. Several methods can be used to allocate less significant parts of audio words to data at fractional bit rates.

One method conveys $\log_2 M$ bits for integer M in the less significant parts of audio words by conveying data in the M possible values of the remainder of the integer audio word after division by M, whereas the rounding quantization process used for the audio involves rounding to the nearest multiple of M. For M a power of 2, this reduces to conventional quantization to $\log_2 M$ fewer bits.

In Eqs. (2-3) and (2-4) above, we described how such M-level data channels can be pseudo-randomized by pseudo-random congruence encoding and decoding. Alternatively, if M can be expressed as nontrivial product of K = two or more integer factors $M = \prod_{j=1}^{K} M_j$, then one can uniquely expand the M-level data word w in the form

$$w = \sum_{k=0}^{K-1} w_{(k)} \prod_{j=1}^{k} M_j \tag{2-7}$$

with $w_{(k)}$ an integer between 0 and $M_{k+1}-1$. Eq. (2-7) is the generalization of the expansion of a number to base M_0 in the case $M_j = M_0$ for all $j = 1,\ldots,K$. Each of the expansion coefficients $w_{(k)}$ can, if desired, be separately pseudo-randomized before the final length M word is formed. Again, this generalizes the binary case described above where the M_j's equaled 2.

A second method for fractional bit rates especially suitable for very low data rates of 1/q BPSS for integer q is to code data only in one out of every q audio samples. The encoding schemes are as before but with a data sampling rate divided by q, and decoding involves the decoder trying out and attempting to decode each of the q possible sub-sequences until it finds out (e.g. by confirming a parity check encoded into the data) which one carries data.

For integers $p < q$, a data rate of p/q BPSS can similarly be obtained by encoding data in the LSBs of p out of every q samples (for example, samples 1 and 3 out of every successive 5 samples for $p = 2$ and $q = 5$).

A third method for fractional bit rates also codes data in the LSBs of q successive samples, but codes the data into different logical combinations of all q bits. For example, a data rate of 1/q BPSS can be obtained by encoding data as the parity (Boolean sum) of the q LSBs. It turns out that this option is often capable of significantly less audio noise degradation than the simpler scheme of the second method. A part of the advantage is that if one needs to modify the parity, then one can choose to modify that sample out of the q successive samples causing the least error in an original high-resolution audio signal, rather than being forced to alter a fixed sample.

We shall see in the following that, for all three kinds of fractional bit rate data encoding, it is possible to use a subtractive dithering technique by a data noise signal to eliminate unwanted modulation noise and distortion side effects on the modified waveform data. The advantages of the new process are not confined to integer bit rates per sample.

3. Subtractively dithered noise shaping

3.1 Subtractive dither

Here we briefly review the ideas of subtractively dithered noise shaping, detailed by the authors in refs. [1], [3] and [4]. In this paper, by a "quantizer" we mean a signal rounding operation that takes higher resolution audio words and rounds them off to the nearest available level at a lower resolution. We assume that the quantizer is uniform, i.e. that the available quantization levels are evenly spaced, with a spacing or step size denoted as STEP.

The quantizer rounding process introduces nonlinear distortion, but this distortion may be replaced by a benign white noise error at the same typical noise level by using the process of subtractive dither shown in figure 4. The process comprises adding a dither noise before the quantizer and subtracting the same dither noise afterwards. Provided that the statistics of the dither noise are suitable, it can be shown (see [1], [2]) that this results in the elimination of all correlations between the error signal across the subtractively dithered quantizer and the input signal. One such suitable dither statistics is what we term RPDF dither, i.e. dither each of whose samples is statistically independent of other samples and with a rectangular probability distribution function with peak levels ±½ STEP.

An audio word of B bits each of which is a pseudo-random binary sequence, is a 2^B-level approximation to a signal with RPDF statistics, so that the data noise signals considered above may be used as dither signals for dithering audio to eliminate nonlinear quantization distortions and modulation noise. Similarly, the M-level data noise signals described above in section 2.3 using the remainder modulo M for data, if made to be of a pseudo-random form by a pseudo-random data encoding/decoding process, can be used as an M-level approximation to RPDF noise.

Although data noise signals are discrete approximations to RPDF noise, they can be converted to continuous RPDF noise statistics by the simple process of adding to them an additional smaller RPDF noise with peak levels ±½ LSB, where LSB is the step size of the LSB's of the transmitted audio words (as distinct from the step size STEP = M LSBs of any rounding process used in encoding hidden data channels.) This is shown schematically in figure 5.

Conventionally, as described in refs. [1] and [3], the use of subtractive dither requires the use of a decoding process in which during playback, the original dither noise added before the quantizer is reconstructed before being subtracted; this requires either the use of synchronized pseudo-random dither generation algorithms, or an encode/decode process in which the dither noise is generated from the LSB's of previous samples of the audio signal [3]. However, in the application of this paper, as will be seen, no special dither reconstruction process is required for the discrete dither, since this is already present in the transmitted LSBs.

3.2 Noise shaping

A white error spectrum is not subjectively optimum for audio signals, where it is preferred to weight the error spectrum to match the ears' sensitivity to different frequencies so as to minimize the audibility or perceptual nuisance of the error. The spectrum of the error signal may be modified to match any desired psychoacoustic criteria by the process of noise shaping, discussed for example in refs. [1], [4], [18]-[20].

Noise shaping may be static (i.e. adjusting the spectrum in a time-invariant way) and made to minimize audibility or optimize perceptual quality at low noise levels, or alternatively it can be made adaptive to the audio signal spectrum so as to be optimally masked by the instantaneous masking thresholds of audio signals at a higher level. The latter option is particularly valuable in the present application, where loud audio signals may well allow an increased error energy to be masked, thereby allowing a higher data rate to be transmitted in the hidden data channels during loud audio passages.

The form of noise shaping with subtractive dither used in this paper is indicated in the schematic of figure 6. It will be noted that, while it is equivalent to some of the forms described in ref. [1], it is not the arrangement described previously by the authors in ref. [3], in that here we put the noise shaping loop around the whole subtractive process. With the arrangement of figure 6, the output of the quantizer itself differs from the noise shaped output of the whole system by a spectrally white dither noise, so that in this arrangement, unlike those suggested in ref. [3], the spectral shape of the quantizer output error and system output error is not identical.

With the noise-shaped subtractively dithered quantizer of fig. 6, the error feedback filter $H(z^{-1})$ must include a 1-sample delay factor z^{-1} in order to be implementable recursively, and the originally white spectrum of the subtractively dithered quantizer is filtered by the frequency response of the noise shaping filter

$$1 - H(z^{-1}), \qquad (3\text{-}1)$$

which is preferably chosen to be minimum phase to minimize noise energy for a given spectral shape [1], and may be chosen to be of any desired spectral shape.

Other implementations of noise shaping around a dithered quantizer system are possible. Alternative implementations are reviewed in ref. [4]. By way of example, fig. 7 shows an alternative "outer" form of noise shaping architecture described in ref. [4], that is equivalent to fig. 6 if one puts

$$H'(z^{-1}) = H(z^{-1})/(1-H(z^{-1})). \qquad (3\text{-}2)$$

The application of noise shaping around a subtractively dithered quantizer will not result in any unwanted nonlinear distortion or modulation noise, provided that the dither noise added in figs. 6 or 7 is RPDF dither matched to the step size STEP of the quantizer.

4. Application to buried data channels

4.1 Noise-shaped subtractively dithered buried channel encoding

Either the arrangement of fig. 6 or fig. 7 can be applied to obtain subtractively dithered noise-shaped audio results when the last digits of an audio signal word (whether the last N binary digits or the remainder after division by M) are replaced by buried data bits.

The procedure is now simple to describe. First the data is pseudo randomized, and then used to form a data noise signal as described above. This data noise signal has (discrete M-level) RPDF statistics, and may be used as the dither noise source in figures 6 or 7, as shown in figs 8 and 9, where the quantizer is simply the process of rounding the signal word to the nearest integer multiple of M LSB's (or the nearest level if the levels are placed uniformly at other than the integer multiple of M LSB's). The process shown in figures 8 or 9 subtracts the data noise signal from the audio at the input of the uniform quantizer (which has step size STEP = M LSBs), and adds it back again at the output of the quantizer so as to make the least significant digits of the output audio word equal to the data noise signal. Noise shaping is performed around this whole process.

For best results using the algorithms of figs. 8 or 9 (or equivalent algorithms such as that in figure 10 below), it is best if the input audio word signal is available at a higher resolution or wordlength than that used in the output, since this will avoid cascading the rounding process used in figs. 8 or 9 with another earlier rounding process. By making the input signal available at the highest possible resolution, any overall degradation of signal-to-noise ratio is minimized.

Since the output equals the output of the quantizer plus the data noise signal, the noise shaping has no effect on the information representing the data in the output audio word, but merely modifies the process by which the quantization of the audio is performed so as to minimize the perceptual effect of the added data noise on the audio. It is remarkable that this output signal, being the output of a noise-shaped subtractively dithered quantizer, automatically incorporates all the benefits of noise shaped subtractive dither without the audio-only listener needing any special subtractive decoding apparatus.

Moreover, because the information received by the data-channel user is not dependent on the noise shaping process, the noise shaping can be varied in any way desired without affecting reception of the data (provided only that no overflow occurs in the noise shaping loop near peak audio levels - fitting a clipper in the signal path before the quantizer to prevent this may be desirable). Thus the noise-shaping process does not affect the way the signal is used by either audio or data end-users of the signal, and so does not need any standardization, but may be used in any way desired by the encoding operative to achieve any desired kind of static or dynamic noise shaping characteristic.

Other equivalent noise-shaped dithering architectures may be used in place of those shown in figs. 8 and 9 for encoding the data signals into the output audio word, using the kind of equivalent architectures discussed in ref. [4]. Purely by way of example, fig. 10 shows yet another implementation having identical performance to that shown in figs. 8 or 9. It is also evident that in a similar way, the data noise signal can be added and subtracted outside the "outer" noise shaper of fig. 9 rather than inside the noise shaper as shown.

4.2 Buried Channel Decoding

Optimum recovery of the audio channels involves no need for any kind of decoder in this proposal. Playback is conventional, with the effect of subtractive dither by the data noise signal being automatic as described above.

Recovery of the buried data is also straightforward, simply being recovery of the data noise signal by rejecting highest bits of the received audio word, or in the case of M-level data, the inverse process to the encoding, of reading the remainder of the audio word after division by M, i.e. resolving the least significant digits of the audio word via modulo M arithmetic. This is followed by the inverse pseudo-random decoding process to recover the data before pseudo randomization, and then the data is handled as data in the usual way.

This decoding process is shown schematically in figure 11.

In the case that the data is encoded as integer coefficients $w_{(k)}$ with more than one base M_j as in Eq. (2-7) above, the data is recovered by K successive divisions by M_1 to M_K, at each stage discarding the fractional part, the K coefficients $w_{(k)}$ being the integer remainders of the division by M_{k+1}. This is the same process shown in figure 11, but with K stages of the modulo division.

5. Vector quantization and dither

5.1 Reasons for digression

It may not be completely clear to the reader without further explanation that the above descriptions of the use of noise shaped subtractive dithering also apply to the stereo parity coding case as well. To see this, we need first to look at vector quantization and vector dithering, and show that exactly the same ideas for subtractive dithering, noise shaping and data encoding can be applied to the vector quantizer case as the scalar case described above. Because the description in this section (section 5) may be found rather technical, we suggest that it be omitted on first reading.

The description here is given in greater generality than needed just for the stereo parity coding case, since it has applications to coding information in the parity of the corresponding bits in 3 or more channels in transmission media carrying more than two audio or image channels, for example in the 3 channels containing the 3 components of a color image.

5.2 Uniform vector quantizers

As briefly indicated in earlier papers [1], [3], [9], the concepts of additive and subtractive dither can be applied to vector as well as scalar quantizers. Vector quantizers quantize a vector signal $\boldsymbol{y}$ comprising n scalar signals $(y_1,\ldots,y_n)$ in geometrical regions covering the n-dimensional space of n real variables. As in the scalar case, we shall say that a vector quantizer Q is a uniform quantizer if the signal y is quantized to a point in a discrete grid G of quantization vectors $\{\boldsymbol{y}_g : g \in G\}$, where there exists a region $\boldsymbol{C}$ around $(0,\ldots,0)$ of n-dimensional space such that the regions $\boldsymbol{y}_g + \boldsymbol{C} = \{\boldsymbol{y}_g + \boldsymbol{c} : \boldsymbol{c} \in \boldsymbol{C}\}$ cover without overlap (except at their boundary surfaces) the range of signal variables y being quantized. Thus a uniform vector quantizer divides the n-variable space into a grid of identical vector quantization cells that are translates of the cell $\boldsymbol{C}$ to the points of the grid G, and quantizes or rounds any point in the cell $\boldsymbol{y}_g + \boldsymbol{C}$ to the point $\boldsymbol{y}_g$.

There are many examples of uniform vector quantizers, the simplest of which has a hypercubic cell $\boldsymbol{C}$ = the region $\{(c_1,\ldots, c_n) : |c_i| \leq \frac{1}{2}\,\mathrm{STEP}\ \forall\ i = 1,\ldots,n\}$, i.e. separate scalar quantization of the n variables. The grid G in this case is simply points of the form $(m_1\mathrm{STEP}, m_2\mathrm{STEP},\ldots,m_n\mathrm{STEP})$ for integer m_j's, and the associated vector quantizer is simply that that takes $(y_1,...,y_n)$ to m_j = integer(y_j/STEP) for j = 1,...,n. This case is trivial in the sense that it is equivalent to using separate uniform scalar quantizers on each of the n channels.

A more complicated but easily visualized example is the 2-channel case where **C** is a regular hexagon in the plane, for example the region consisting of the points (c_1,c_2) in the plane such that

$$|c_1| \le ½\ \mathrm{STEP},\ |-½c_1 + (\sqrt{3}/2)c_2| \le ½\ \mathrm{STEP},\ |-½c_1 - (\sqrt{3}/2)c_2| \le ½\ \mathrm{STEP}, \qquad (5\text{-}1)$$

and the grid G is the centers of the hexagons in the honeycomb grid covering the plane, i.e. G is the points

$$((m_1+½m_2)\mathrm{STEP},\ (\sqrt{3}/2)m_2\mathrm{STEP}) \qquad (5\text{-}2)$$

for integer m_1 and m_2.

A uniform vector quantizer of particular interest and practical use in n dimensions is what we shall term the rhombic quantizer. This starts off with a conventional hypercubic grid G_C of points at positions (m_1STEP,m_2STEP,...,m_nSTEP), where STEP is a step size, and m_1 to m_n are integers, which of course has the hypercube quantizer cell described just above and corresponds to the use of n separate scalar uniform quantizers. However, we the produce a new grid $G \subset G_C$ which consists of just those grid points in G_C with $m_1+...+m_n$ having even integer values. This new grid only has half as many points as the original, and can be equipped with a new vector quantization cell **C** as follows, which we shall term the n-dimensional rhombic quantizer cell.

The rhombic quantizer cell can be described geometrically by thinking of the original hypercubic cells as being colored white if $m_1+...+m_n$ is even and black if $m_1+...+m_n$ is odd, forming a kind of n-dimensional checkerboard pattern of alternately black and white hypercubes. Then attach to each white hypercube that "pyramid" portion of each adjacent black hypercube lying between the center of the black hypercube and the common "face" with the white hypercube. The resulting solid is the rhombic cell ***C***.

It is evident, since the pyramid portions taken from adjacent black hypercubes are in total enough to form one black hypercube if pieced together, that the volume occupied by the rhombic quantizer cell is twice that occupied by the original hypercube quantizer cell, and that the versions of the rhombic quantizer cell translated by the grid G indeed cover the n-dimensional n-parameter vector signal space.

For n = 2, the rhombic quantizer cell ***C*** is a diamond-shape, being a square whose sides are rotated 45° relative to the channel axes, as shown in fig.12. For n = 3, the rhombic quantizer cell ***C*** is a rhombidodecahedron, a 12-faced solid whose faces are rhombuses. For n = 4, the rhombic quantizer cell ***C*** is a regular polytope unique to 4 dimensions termed the regular 24-hedroid [23].

Calculations involving quite complicated multidimensional integrals, which we shall not detail here, show, for a given large number of quantizer cells covering a large region of n-dimensional space, that for $n = 2$, rhombic quantization has the same signal-to-noise ratio (S/N) as conventional independent quantization of the channels, but that for $n \geq 3$, rhombic quantizers give a better S/N than conventional independent quantization of the channels. The improvement reaches a maximum of about 0.43 dB when $n = 6$. This improvement in the S/N is maintained when additive or subtractive dither is used as described below. (The hexagonal 2-channel quantization described above gives a 0.16 dB better S/N than independent quantization of 2 channels.)

Mathematically, the rhombic quantizer has grid G consisting of the points

$$(m_1\mathrm{STEP}, m_2\mathrm{STEP}, \ldots, m_n\mathrm{STEP}), \qquad (5\text{-}3a)$$

where the m_i have integer values with

$$m_1 + \ldots + m_n \text{ having even integer values.} \qquad (5\text{-}3b)$$

The rhombic cell **C** is that region of points $(c_1, \ldots, c_n)$ satisfying the $n(n-1)$ inequalities

$$|c_i + c_j| \leq \mathrm{STEP}, \; |c_i - c_j| \leq \mathrm{STEP}, \qquad (5\text{-}4)$$

for $i \neq j$ selected from $1, \ldots, n$. The associated uniform vector quantizer rounds a vector signal $(y_1, \ldots, y_n)$ by an algorithm whose outline form might be

```
        m'_i := integer(y_i/STEP),
        If m'_1+...+m'_n is even
                then m_i := m'_i for all i = 1,...,n,
                else   c_i := y_i/STEP - m'_i,
        (*)             d_j : = sgn(c_j) if |c_j| > |c_i| for all i < j and |c_j| ≥ |c_i| for all i > j
                        d_i := 0 for all other i,
                        m_i := m'_i + d_i for all i = 1,...,n.
                End If                                                      (5-5)
```

There are, of course, various equivalent forms for this kind of rhombic quantizer algorithm, a computationally demanding aspect on typical signal processors being the determination in line (*) of that j for which $|c_j|$ is biggest.

In the $n = 2$ case, there is a simpler rhombic quantization algorithm as follows

$$x_1 := y_1 + y_2, \; x_2 := y_1 - y_2,$$
$$m'_1 := \mathrm{integer}(x_1/((\sqrt{2})\mathrm{STEP}))$$
$$m'_2 := \mathrm{integer}(x_2/((\sqrt{2})\mathrm{STEP}))$$
$$m_1 := m'_1 + m'_2, \; m_2 := m'_1 - m'_2, \qquad (5\text{-}6)$$

which is based on the observation that the rhombic quantizer cell for $n = 2$ is the same shape as the square cell used for ordinary independent quantization of the two channels, but rotated by 45° and with an increase of the step size by a factor $\sqrt{2}$. (See fig. 12).

5.3 Subtractive vector dither

The concepts of dithering for uniform quantizers developed in refs. [1-4] for scalar uniform quantizers may be applied also to the vector case by using appropriate vector dithers. An n-signal dither noise vector $(v_1, \ldots, v_n)$ is said to have uniform probability distribution function in a region **C** of n-dimensional space if its joint probability distribution function is constant within the region **C** and zero outside it. This is the n-dimensional generalization of rectangular PDF dither for vector signals, and we denote the associated n-vector dither signal by $\mathbf{r_C}$.

It can be shown (we omit any proofs here) that if the subtractive dither arrangement of figure 4 is used for modifying an input vector signal, where the "uniform quantizer" becomes a vector uniform quantizer with quantization cell **C**, and the dither noise becomes a uniform PDF vector dither $\mathbf{r_C}$ on the region **C**, then the output vector signal of the system is free of all nonlinear distortion and modulation noise effects (i.e. the first moment of the output signal error is zero, and the second moment independent of the input signal [4]). Moreover, this is still the case if any statistically independent additional noise is added to the uniform PDF dither noise $\mathbf{r_C}$ on the region **C**.

Moreover, noise shaping can be applied around such subtractive dither in exactly the same way as before, as shown in figs. 6 and 7, or in equivalent noise shaping architectures, the only difference being that any filtering is now applied to n parallel signal channels. It is also possible, if desired, to use an nxn matrix error feedback filter $H(z^{-1})$ or $H'(z^{-1})$ in order to make the noise shaping dependent on the vector direction, for example to optimize directional masking of noise by signals [9], [10].

It is possible to generate uniform PDF vector dither $\mathbf{r_C}$ over the rhombic cell **C** described above by an algorithm such as the following: First generate, for example by the well-known congruence method, n statistically independent rectangular PDF dither signals r_i ($i = 1,\ldots,n$) with peak values ±½ STEP, and also generate an additional two-valued random or pseudorandom signal u with value either 0 or 1. Then the values of the noise signal $\mathbf{r_C} = (v_1,\ldots,v_n)$ are given by:

If $u = 0$
    then $v_i := r_i$ for all $i = 1,\ldots,n$,
    else $d_j := \mathrm{sgn}(r_j)$ if $|r_j| > |r_i|$ for all $i < j$ and $|r_j| \geq |r_i|$ for all $i > j$
        $d_i := 0$ for all other i,
        $v_i := r_i - d_i\mathrm{STEP}$ for all $i = 1,\ldots,n$.
End If. (5-7)

However, in applications of subtractive dither, this algorithm may involve unnecessary complication, since it can be shown that with the subtractive dither arrangement of fig. 4 with a uniform vector quantizer with quantization cell **C**, that a uniform PDF vector dither signal $\mathbf{r_D}$ may be used for *any other* uniform quantization cell **D** sharing the same grid G, and still will eliminate nonlinear distortion and modulation noise in the output. Whatever the shape of the other quantization cell **D** used for the dither signal, the resulting error signal from the subtractive dither arrangement of fig. 4 is a noise signal with uniform PDF statistics on the quantizer cell **C** of the uniform vector quantizer used.

This can allow a much simpler algorithm to be used for generating the vector dither in which uSTEP is added to (or subtracted from) just one of the n rectangular PDF noise components. For example, a uniform PDF vector dither noise signal $\mathbf{r_D} = (v_1,\ldots,v_n)$ given by

$$v_1 := r_1 - \text{uSTEP}$$
$$v_i := r_i \text{ for } i = 2,\ldots,n. \qquad (5\text{-}8)$$

may be used to subtractively dither the above rhombic quantizer.

5.4 Nonsubtractive case

Although we shall not need to use the nonsubtractive vector dither case in the hidden data channel application of this paper, it is easy to note the extension of the above to the nonsubtractive case. As in the scalar case reported in ref. [2], it can be shown that a uniform vector quantizer with quantizer cell ***C*** can be made to give an output suffering from no nonlinear distortion or modulation noise if dither noise is added before the quantizer that has the form of the sum of two statistically independent uniform PDF vector dithers each of the form $\mathbf{r_C}$ over the region ***C***.

Such a dither is a vector analog of the triangular PDF dither [2] used in the scalar case, and may similarly be subjected to noise shaping of the dithered vector quantizer without introducing nonlinear distortion or modulation noise effects. As in the scalar case, such nonsubtractive dithering with no modulation noise gives a noise energy 3 times as large as does subtractive dithering.

6. Refinements of the basic proposal.

6.1 Further developments

The encoding process described above will work well as it stands, but does not incorporate various desirable refinements which we shall now describe. These include methods to take account of the fact that the data noise signal has a discrete and not a continuous PDF dither, and applications involving stereo parity coding.

6.2 Non-discrete dither

The fact that the dither given by the data noise signal has an M-level discrete probability distribution function rather than a continuous RPDF means that there is still unwanted quantization distortion at the level of the LSB of the audio word which is not properly dithered. Preferred methods of adding "non-discrete" dither (or, strictly speaking, dither at a significantly high arithmetic accuracy such as implemented using 24 or 32 bit arithmetic) are now described. The method of adding such dither shown in fig. 5 is not preferred for three reasons:

(i) Optimum playback requires subtractive decoding of the ±½ LSB RPDF dither signal, with all the usual problems of implementing subtractive dither [1], since unlike the discrete data noise signal, this is not explicitly transmitted in the audio word.

(ii) the ±½ LSB RPDF dither signal added before the quantizer does not eliminate modulation noise in non-subtractive playback, having the wrong statistics for this purpose [2], and

(iii) if the whole system is noise shaped as in figs. 6 or 7, the nonsubtractive listener will hear the ±½ LSB RPDF dither signal as having a white spectrum not affected by the noise shaping, so will perceive an increase in noise level.

A correct way of adding extra dither to avoid nonlinear quantization distortion and modulation noise at the ±½ LSB level is shown in figure 13. The dither used has a triangular PDF with peak levels ±1 LSB (so-called TPDF dither) with independent statistics at each discrete time instant, so as to eliminate modulation noise in nonsubtractive playback [2], and is added before the quantizer in the noise shaping loop, but not subtracted in the noise shaping loop. This ensures that the added noise in nonsubtractive playback is noise shaped.

Subtractive playback of the extra dither is done, also as shown in fig. 13 by reconstituting the triangular ±1 LSB PDF dither at the playback stage, passing it through a noise shaping filter $1 - H(z^{-1})$, and subtracting the filtered noise from the output audio word. Subtractive playback of course reduces the extra noise energy caused by the non-discrete dither by a factor 3, although this will only be highly advantageous in the case that the data noise signal has fairly low energy, e.g. at a data rate of 1 BPSS.

The triangular dither signal may be generated, in encoding, as proposed in the "autodither" proposal of ref. [3] by means of a pseudo-random logic look-up table (or a logic network having the effect of a pseudo-random look-up table) from the less or least significant parts of the *output* audio word in the last K previous samples, where typically K may be 24, and can be reconstructed from the same audio word at the input of the system by the same look-up table or logic in the decoding stage. This is shown in the case of the system of fig. 13 in fig. 14.

Although figures 13 and 14 are shown for the particular noise shaping architecture of fig. 6, similar ways of adding the extra triangular dither can be used with any other equivalent noise shaping architecture such as the outer form of figure 7 and fig. 10 - again by adding the triangular dither just before the quantizer and subtracting it again, via a noise shaping filter $1 - H(z^{-1})$, only at the output of the decoder. It is clear that the points at which dither signals are added can be shifted around in various ways without affecting the functionality.

6.3 The stereo parity case

Suppose we have 2-channel stereo signals in which data is encoded pseudorandomly in bit N for all N = 15 to say 15-h+1 (where the integer h may typically be any integer from say 0 to perhaps 6 or 8, the case 0 being the case of no bits being encoded) of the left and right audio words, and data also being encoded in the stereo parity (Boolean sum) of bit 15-h of the left and right audio words, as described in subsection 2.2 above.

Based on the results on uniform vector quantization and subtractive vector dither of section 5 above, the noise-shaped subtractive encoding of the data described above in the scalar case for individual audio channels may be applied to this case too with just two reinterpretations of the above:

(i)The uniform quantizer used in figs.6-10 now becomes a uniform 2-dimensional rhombic quantizer (such as described in Algorithm (5-6) and illustrated in fig. 12) with STEP = 2^h LSB.

(ii) the "data noise signal" used for dithering is given, for example, by Eqs. (5-8) where r_i is the data noise signal of the last h bits of the i'th-channel audio word (with the first channel being say left and the 2nd channel being say right), and u being the parity of bits 15-h of the left and right audio words. In units of LSB, the data noise signal for the left channel is then $L_0 - 2^h u$ and for the right channel is R_0, where L_0 and R_0 are the respective integer words represented by the last h bits of the audio word formed by the data in the two channels.

Any alternative data noise signal may be used that represents an appropriate uniform PDF vector dither as described in section 5.3, such as for example that given by Algorithm (5-5).

The residual nonlinear distortion and modulation noise effects at the LSB level caused by the fact that the vector data noise is discrete rather than continuous can be removed by using exactly the same technique described in subsection 6.2 and figs. 13 and 14 above by adding and, where appropriate, subtracting ±1LSB triangular PDF dither in each channel separately, the only difference being that the uniform quantizer has become a rhombic vector quantizer and the data noise signal has a modified vector form as just described.

The particular case h = 0, where data is transmitted only in the parity of the LSB of the audio word in 2 channels, simply uses the parity signal itself at the LSB level as a "data noise signal" in one of the two channels in the encoding process - it does not matter which of the two channels is chosen. With subtractively dithered playback, it turns out that the use of properly designed stereo parity coding of data, using a rhombic vector quantizer in the encoding process, gives a total noise level 1 dB lower than would the process of coding the data into the LSBs of the words of just one of the two audio channels. Thus stereo parity coding at low bit rates not only ensures audio left/right symmetry for added noise, but gives a significant noise level advantage.

6.4 Generalized stereo parity coding

There are various generalizations of the particular stereo parity coding case just described. We outline these briefly to show the applicability of these ideas to other cases.

A first obvious generalization is that obviously the same process may be applied to other audio wordlengths besides the 16 bit wordlength of CD - for example to the 10 bit wordlengths of NICAM encoded digital signals or to the 20 bit or 24bit wordlengths used in some professional audio applications when it is desired to hide data in the audio words. For example, in ref. [3], the authors described a proposal to add data at the 24th bit in studio operations on signals to detect whether or not they had been modified, and the data encoding techniques of this paper can be used in that application to minimize the audibility of the modification of the signal proposed there.

The second generalization is that one can also apply stereo parity coding to the case where one replaces the 2^h-level data in the last h bits by an M-level case for any integer $M > 1$. In this case, data is coded into the residue of the audio words of the two channels after division by M, and the "stereo parity" data channel is coded into the Boolean sum of the binary LSB in the two channels of the integer parts of the audio words divided by M. This case is handled identically to that in the previous sub-section 6.3 except that 2^h is replaced throughout by M, and the phrase "last h bits" is replaced by "residue modulo M").

A third generalization instead considers n channels rather than two. As before, this uses a rhombic quantizer in the encoding process for STEP = M LSBs, but now the n-dimensional rhombic quantizer described in (5-3) to (5-5) above, and a vector data noise signal comprising the n M-level data noise signals generated for the residue modulo M data conveyed in each of the n audio channels, to just one of which at each instant is added or subtracted uSTEP, where u is the parity (i.e. Boolean sum) of the binary LSB in the n channels of the integer parts of the audio words divided by M. Other than replacing the ordinary uniform quantizers with step size STEP by a rhombic quantizer and using the modified data noise signal, the descriptions given earlier for coding data still apply to this case.

Note that the choice of which channel of the vector data noise signal to add or subtract uSTEP, and the choice of whether to add or subtract, can be made freely, and that this choice can be made adaptively instant by instant to minimize data noise energy if desired, e.g. by making that choice which minimizes the maximum of the data noise signals in the n channels at each instant. This choice is (a discrete approximation to) that described in (5-7) for uniform PDF vector dither over a rhombic quantizer cell.

6.5 Low bit-rate case

If one has n transmitted channels of audio, then the parity of their LSBs can be used to transmit a 1 bit per n-channel-sample data channel, with remarkable little loss of S/N, especially in the case that full subtractive dithering is used at the LSB level. One might expect a loss of S/N of 6.02/n dB because the loss is shared among n channels, but for $n > 2$, one gets a smaller loss, typically between 0.3 and 0.4 dB better, because of the fact noted in section 5 that rhombic vector quantization has a better S/N than independent channel quantization for a given density of quantization points in the quantization grid. For n = 6, a 1 bit per n-channel-sample subtractively dithered buried data channel causes a S/N degradation of under 0.6 dB compared with a properly dithered case with no buried data channel.

Exactly the same techniques can be used to convey data via q successive samples of a monophonic signal, for example by coding into the parity of the LSBs of each successive block of q samples, as described in section 2.3. What we have now shown is that by using the parity signal as a subtractive dither for any one sample with a q-dimensional rhombic quantizer, plus normal triangular additive or subtractive dither, that this fractional rate channel can be coded with a very small loss of S/N (e.g. 0.6dB for a block length q = 6), and yet with no nonlinear distortion or modulation noise in either nonsubtractive or subtractive reproduction.

This kind of efficient low bit-rate culling of data capacity could be used, for example, with successive samples within individual subband channels of a subband data compression system. Its application is not confined to audio; culling say 1 bit per 6 10-bit video samples in a digital video recorder with a video data rate of 200 Megabits per sec. would give a data rate typically enough for 4 16-bit audio channels or a consumer-grade additional data-reduced video signal while losing only 0.6 dB in video S/N in the original video channel.

7. Conclusions

7.1 Audio Quality Considerations

Anyone concerned with the future potential of the audio art will have some concern about using information originally allocated to a high-quality audio signal to transmit other data instead, as in the proposal in this paper. In order to encourage progress in the audio art, there is a need for at least one widely available consumer medium without built-in serious quality compromises, such as CD (unlike data reduced digital systems) offers, so that the market is there in which recordings with improved quality can be made, heard and sold. Without such a medium, we shall find ourselves permanently locked into limitations many of which will only become apparent as the art of recording, psychoacoustics and studio production develop further.

Even the best theoretical models of the ears are still extremely crude, for example not describing the effect of hearing multiple events with individually low but jointly high detection probabilities, especially for non-stationary or transient signals. Many of the musical subtleties of the best "purist" recordings probably reside in these areas of our technical ignorance.

We have therefore been concerned to devise buried channels that satisfy far more stringent requirements than simply satisfying crude masking models, which we feel still have limited applicability to state-of-the art recording quality. This conservative attitude means that (although the option is there with adaptive data rates and noise shaping for our proposal to code data if desired to satisfy existing masking models) such masking models are in no way assumed in the standard. It is a matter of judgment on a case-by-case basis of individual recordings whether such signal manipulations of the error are subjectively acceptable.

In cases where such compromises are not acceptable or are considered too risky (especially for material with high or serious artistic intent), our proposal allows the hidden data channels to produce the most benign kind of error - namely a steady noise error free of all nonlinear distortion or modulation noise, and having any desired spectral shape. Unlike previous proposals, this allows avoidance of all psychoacoustically disturbing patterns in the error signal, whether related to the audio signal or to patterns in the transmitted data.

The beauty of this proposal is that, by incorporating noise shaping and subtractive dither, it avoids adding any more error noise to the audio signal than is strictly necessary to handle the desired data rate, typically allowing up to 20 dB better perceived signal-to-noise ratio than would be achieved simply by replacing the relevant audio word bits by data, and typically allowing up to 25 dB better perceived signal-to-noise ratio than would be achieved were one also to attempt adding dither in a simple replacement scheme to avoid nonlinear distortion and modulation noise.

Particularly at low data rates, the audio performance of our proposed scheme will typically be comparable to or better than some of the better noise shaped dithering systems currently on the market, i.e. a CD carrying the hidden data channels is likely to sound better than current CDs without the data channel, since the encoding standard incorporates properly designed dithering (and optional properly designed noise shaping). Even at the higher data rates, the use of proper dithering may well mean a better sound than is currently the norm.

All other things being equal, an audiophile listener would not choose any degradation of audio quality, even if this takes the form of a smooth steady noise free of unwanted modulation and nonlinear distortion effects. But things are not equal, since the data channels can be used to convey additional audio channels in a fully compatible way. Providing the coding of these additional audio channels is done with sufficient care to avoid audible data reduction artifacts, we believe that the overall improvement obtained by adding at least one extra audio channel, either for horizontal B-format Ambisonic surround-sound or for three-channel frontal stage stereo, may subjectively more than make up for the relatively benign loss of signal-to-noise ratio (compared to the best noise-shaped dithered performance of which CD is capable) of the added data channels.

Alternative audio uses of the additional data channel includes compatible frequency-range extension without the audible degradations of quality heard in existing commercial schemes for this, and the transmission of level-alteration information to allow dynamic range adjustment of the recording for users equipped with data decoders.

7.2 Summary

In this paper, we have described a method of forcing the least significant information in the audio words to conform to the data values of data channels, while ensuring that the effect on the audio is that of adding a noise-shaped steady pattern-free random noise at a level no greater than would be expected from Shannon information theory from the number of bits "stolen" from the audio for an optimally noise shaped subtractively dithered system.

These techniques involve a process for pseudo-randomizing the data so that the audio sees it as a random noise signal which is optimized for subtractively dithering the audio, to eliminate both nonlinear distortion and modulation noise. Not only is the subtractive dithering automatically operative in ordinary playback, but additionally full noise shaping can be applied to the data dither as well.

This paper has further extended this technique not just to the encoding of data in individual audio signals, but to a technique, stereo parity coding, that allows efficient coding of data jointly into two or more audio channels, by using a vector quantization and subtractive vector dithering process. The joint coding process not only ensures symmetry of the way noise is distributed among the audio channels, but additionally gives a substantial improvement in noise performance, especially at low data rates in the data channels. The attainable noise performance approaches the theoretical Shannon limits for the combined Shannon data rate of the audio and buried data channels.

In describing these techniques, a brief account has been given of the generalization of the ordinary theory of subtractive dither to the vector quantizer and vector dither case.

Possible uses of the resulting benign hidden data channels have been described, including additional audio channels for multiloudspeaker stereo or surround sound, audio bandwidth extension, dynamic range control, as well as obvious data applications such as graphics, text/lyrics, copyright, track information, and even data-reduced video.

Unlike previous approaches, no assumptions have been made regarding the masking abilities of the ears - rather the design aim has been to ensure that the only effect on the existing audio of adding data is to cause a minimal increase in steady background noise, ensuring no compromise with other audio virtues of compact disc. If a noise performance comparable to good current CD's is acceptable, this allows data rates of up to 360 kbit/s to be transmitted in the buried data channel, although much more stringent noise requirements can be met at the expense of a reduced data rate.

The authors are open to approaches from concerns wishing to develop particular applications or develop technical standards for uses of the buried data channels

8. References

[1] M.A. Gerzon & P.G. Craven, "Optimal Noise Shaping and Dither of Digital Signals", Preprint 2822 of the 87th Audio Engineering Society Convention, New York (1989 Oct. 18-21)

[2] S.P. Lipshitz, R.A. Wannamaker & J. Vanderkooy, "Quantization and Dither: A Theoretical Survey", J. Audio Eng. Soc., vol. 40 no. 5, pp. 355-375 (1992 May)

[3] P.G. Craven & M.A. Gerzon, "Compatible Improvement of 16-Bit Systems Using Subtractive Dither", Preprint 3356 of the 93rd Audio Engineering Society Convention, San Francisco (1992 Oct. 1-4)

[4] M.A. Gerzon, P.G. Craven, R.J. Wilson & J.R. Stuart, "Psychoacoustic Noise Shaped Improvements in CD and Other Linear Digital Media", Preprint presented at the 94 Audio Engineering Society Convention, Berlin (1993 March.)

[5] W.R. Th. Ten Kate, L.M. Van De Kerkhof & F.F.M. Zijderveld, "A New Surround-Stereo-Surround Coding Technique", J. Audio Eng. Soc., vol. 40 no. 5, pp. 376-383 (1992 May)

[6] M.A. Gerzon, "Hierarchical Transmission System for Multispeaker Stereo", J. Audio Eng. Soc., vol. 40 no. 9, pp. 692-705 (1992 Sept.)

[7] M.A. Gerzon, "Hierarchical System of Surround Sound Transmission for HDTV", Preprint 3339 of the 92nd Audio Engineering Society Convention, Vienna (1992 Mar.)

[8] M.A. Gerzon, "Compatibility of and Conversion Between Multispeaker Systems", Preprint 3405 of the 93rd Audio Engineering Society Convention, San Francisco (1992 Oct. 1-4)

[9] M.A. Gerzon, "Problems of Error-Masking in Audio Data Compression Systems", Preprint 3013 of the 90th Audio Engineering Society Convention, Paris (1991 Feb.)

[10] M.A. Gerzon, "Directional Masking Coders for Multichannel Subband Audio Data Compression", Preprint 3261 of the 92nd Audio Engineering Society Convention, Vienna (1992 Mar.)

[11] M.A. Gerzon, "Problems of Upward and Downward Compatibility in Multichannel Stereo Systems", Preprint 3404 of the 93rd Audio Engineering Society Convention, San Francisco (1992 Oct. 1-4)

[12] M.A. Gerzon, "The Design of Distance Panpots", Preprint 3308 of the 92nd Audio Engineering Society Convention, Vienna (1992 Mar.)

[13] M.A. Gerzon, "Periphony: With-Height Sound Reproduction", J. Audio Eng. Soc., vol. 21, pp. 2-10 (1973 Jan./Feb.)

[14] M.A. Gerzon, "Practical Periphony", Preprint 15__ of the 65th Audio Engineering Society Convention, London (1980 Feb.)

[15] M.A. Gerzon, "Ambisonics in Multichannel Broadcasting and

Video", J. Audio Eng. Soc., vol. 33 no. 11, pp. 859-871 (1985 Nov.)
[16] A.J. Mason, A.K. McParland & N.H.C. Gilchrist, "Unobtrusive Compression of Dynamic Range", Preprint 3433 of the 93rd Audio Engineering Society Convention, San Francisco (1992 Oct. 1-4)
[17] R.A. Wannamaker, S.P. Lipshitz, J. Vanderkooy & J.N. Wright, "A Theory of Non-Subtractive Dither", submitted to IEEE Trans. Sig. Proc. (1991)
[18] S.P. Lipshitz, J. Vanderkooy & R.A. Wannamaker, "Minimally Audible Noise Shaping", J. Audio Eng. Soc., vol. 39 no. 11, pp. 836-852 (1991 Nov.) Corrections to be published in JAES
[19] S.P. Lipshitz R.A. Wannamaker & J. Vanderkooy, "Dithered Noise Shapers and Recursive Digital Filters", to be presented at the 94th Audio Engineering Society Convention, Berlin (1993 Mar.)
[20] J.R. Stuart & R.J. Wilson, "A Search for Efficient Dither for DSP Applications", Preprint 3334 of the 92nd Audio Engineering Society Convention, Vienna (1992 March)
[21] P.A. Regalia, S.K. Mitra, P.P. Vaidyanathan, M.K. Renfors & Y. Nuevo, "Tree-Structured Complementary Filter Banks Using All-Pass Sections", IEEE Trans Circuits & Systems, vol. CAS-34 no. 12, pp. 1470-1484 (1987 Dec.)
[22] R.E. Crochiere & L.R. Rabiner, "Multirate Digital Signal Processing", Prentice-Hall Inc., Englewood Cliffs, New Jersey (1983), especially chapter 7.
[23] H.M. Coxeter, "Regular Polytopes" (2nd Edition), Macmillan
[24] J.R. Emmett, "Buried Data in NICAM transmissions", Preprint 3260 of the 92nd Audio Engineering Society Convention, Vienna (1992 March)

9. PATENT NOTE

The authors have applied for patents on various techniques described in this paper.

10. ACKNOWLEDGEMENTS

The authors acknowledge useful and relevant discussions with many people on topics related to this paper over the years, including Dr. Geoffrey Barton, Dr. Raymond Veldhuis, and J.R. Emmett.

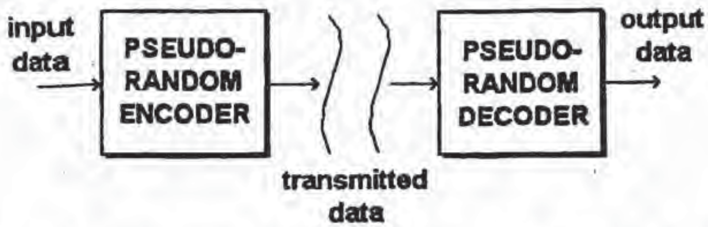


Figure 1. Pseudo random encoding and decoding of data transmitted via CD channel to ensure noise-like behavior.

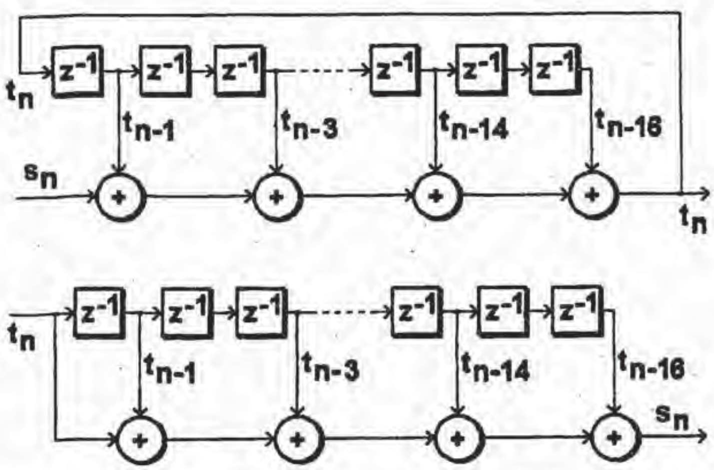


Figure 2. Binary pseudo-random sequence generator using shift-register logic, with input "exclusive or" gate for encoding and decoding of binary data stream.

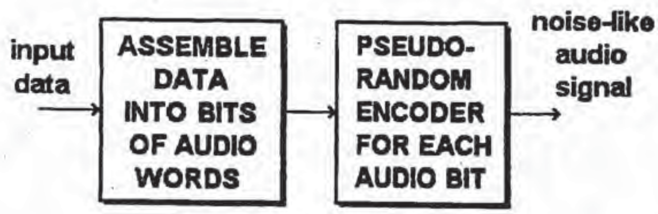


Figure 3. Schematic of processing of data to form audio noise-like signal.

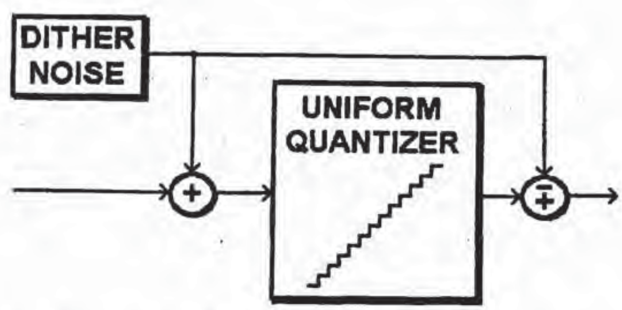


Figure 4. Subtractive dither around a uniform quantizer.

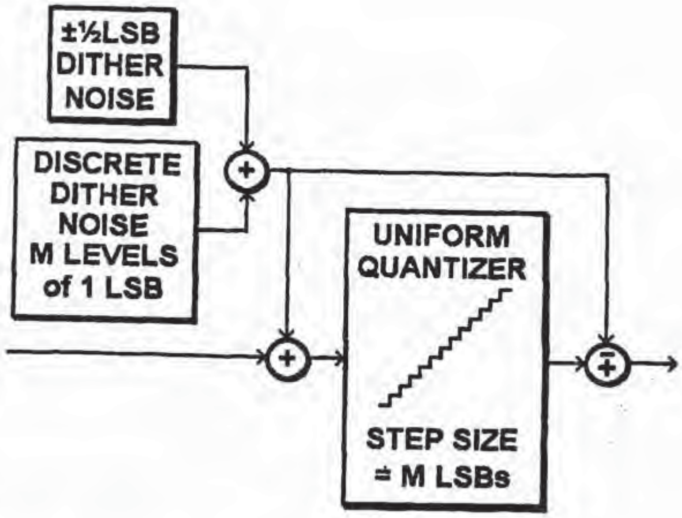


Figure 5. Subtractive dither using a combination of discrete and continuous RPDF dither.

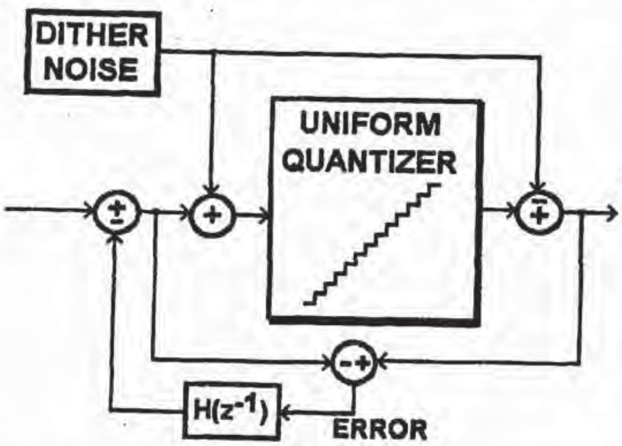


Figure 6. Noise shaped subtractively dithered uniform quantizer.

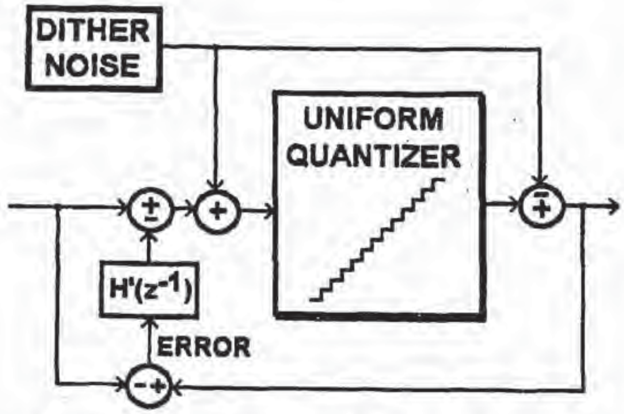


Figure 7. "Outer" form equivalent to that of fig. 6 for noise-shaped subtractively dithered uniform quantizer, where $H'(z^{-1}) = H(z^{-1})/(1-H(z^{-1}))$.

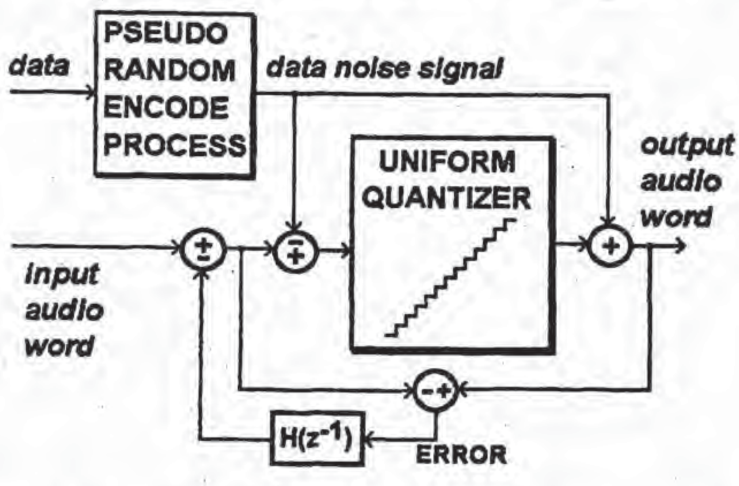


Figure 8. Noise shaping round pseudo random data noise signal encoding of data into an audio word. Standard noise shaper form.

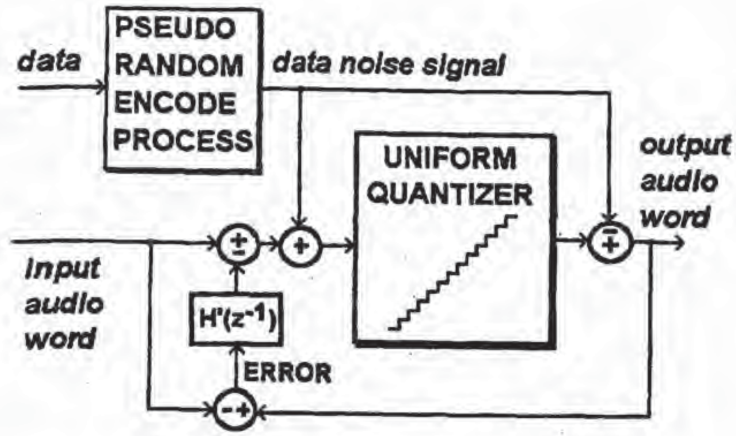


Figure 9. Noise shaping round pseudo random data noise signal encoding of data into an audio word. "Outer" noise shaper form equivalent to fig 8 if $H'(z^{-1}) = H(z^{-1})/(1-H(z^{-1}))$.

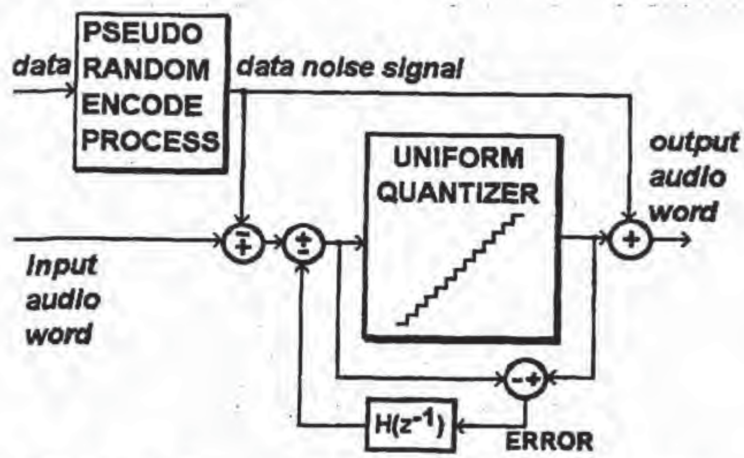


Figure 10. Further implementation of noise shaping round pseudo random data noise signal encoding of data into an audio word.

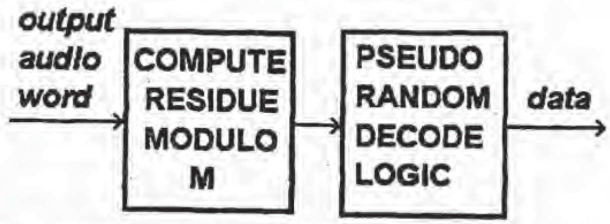


Figure 11. Recovery of the data signal from the received coded audio word.

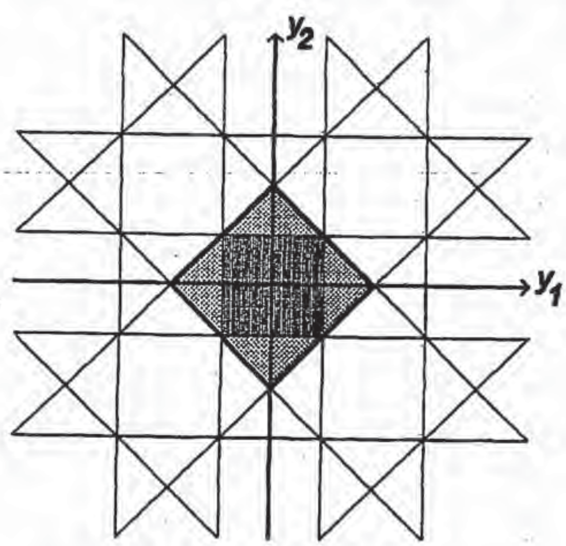


Figure 12. 2-dimensional rhombic quantizer region (shaded square with sides tilted 45°) shown against a background (squares with horizontal and vertical sides) of conventional independent quantizers (whose square quantizer region is darkly shaded) on each channel y_1 and y_2.

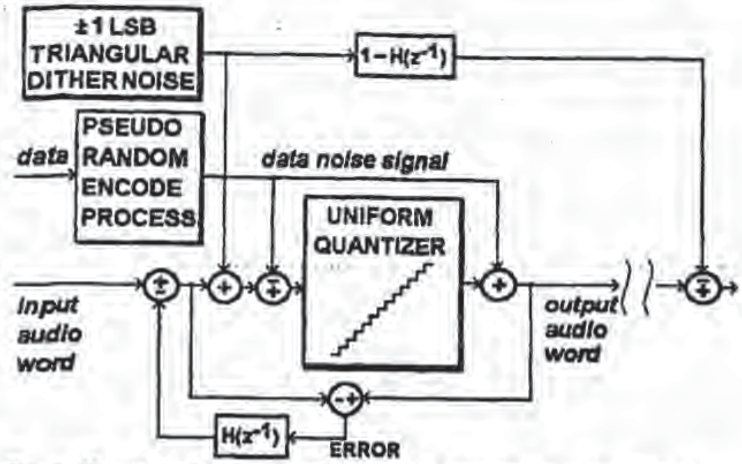


Figure 13. Use of extra subtractive dither to eliminate nonlinear distortion and modulation noise at LSB level, using noise shaped triangular PDF dither having ±1 LSB peaks to achieve good results in both nonsubtractive reproduction of output audio word and (shown) subtractive reproduction.

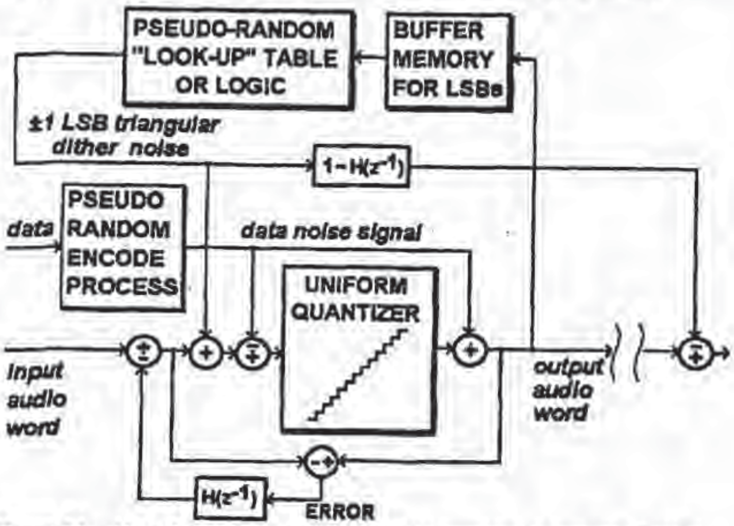
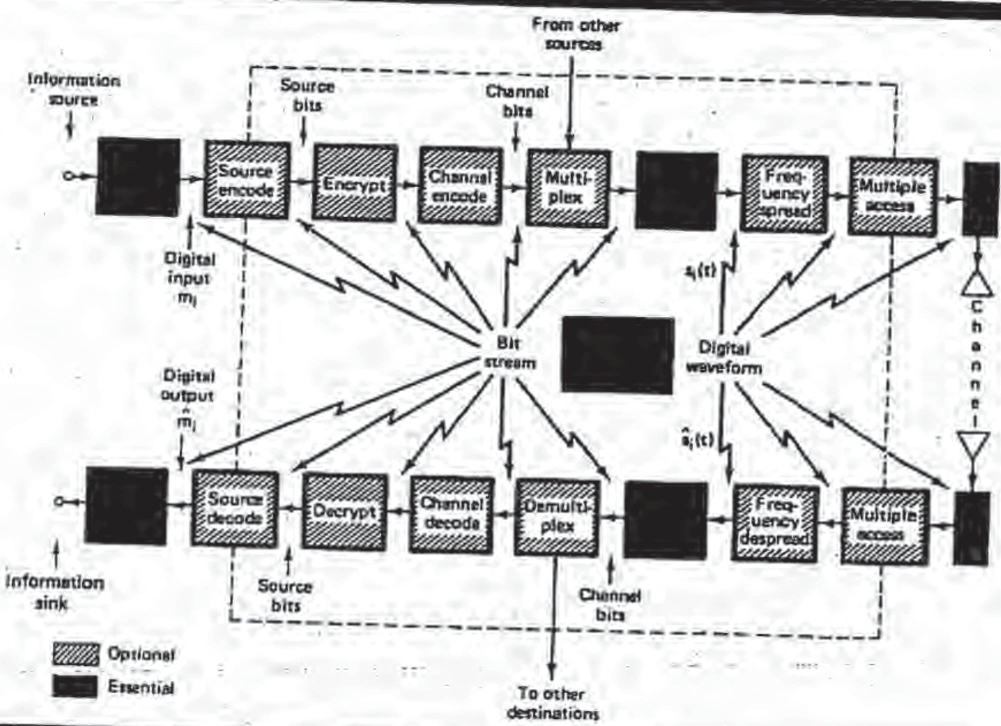


Figure 14. Use of autodither with figure 13 to generate triangular dither in encoder and decoder.

DIGITAL COMMUNICATIONS
Fundamentals and Applications



Library of Congress Cataloging-in-Publication Data

Sklar, Bernard (date)
Digital communications.

Bibliography: p.
Includes index.
1. Digital communications. I. Title.
TK5103.7.S55 1988 621.38'0413 87-1316
ISBN 0-13-211939-0

Editorial/production supervision and
interior design: Reynold Rieger
Cover design: Wanda Lubelska Design
Manufacturing buyers: Gordon Osbourne and Paula Benevento

Printed in the United States of America

20

ISBN 0-13-211939-0

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Pearson Education Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

descriptions of the source (0.470, 0.412, and 0.408 bit, respectively) are decreasing asymptotically toward the source entropy of 0.357 bit/input symbol. Remember that the source entropy is the lower bound in bits per input symbol for this (infinite memory) alphabet and this bound can only be approached asymptotically with finite-length coding.

11.1.2 Waveform Sources

A waveform source is a random process of some independent variable. We classically consider this variable to be time, so that the waveform of interest is a time-varying waveform. Important examples of time-varying waveforms are the outputs of transducers used in process control, such as temperature, pressure, velocity, displacement, and flow rates. Examples of particularly high interest include speech and music waveforms. The waveform can also be a function of one or more spatial variables (e.g., displacement). Important examples of spatial waveforms include single images such as a photograph, or moving images such as the successive images (at 24 frames/s) of moving picture film. Spatial images are often converted to time-varying functions by a simple scanning operation. This, for example, is done for facsimile transmission and with a slight modification (called interlacing) for standard broadcast television.

11.1.2.1 Amplitude Density Functions

Discrete sources were described by a list of their possible elements (called letters of an alphabet) and their multidimensional probability density functions (pdfs) of all orders. By analogy, waveform sources are similarly described in terms of their probability density functions as well as parameters and functions derived from these functions. We model many waveforms as random processes with classical probability distribution functions and with simple correlation properties. In the modeling process we distinguish between short-term or local (time) characteristics and long-term or global characteristics. This partition is necessary because many waveforms are nonstationary.

The probability density function of the actual process may not be available to the system designer. Sample density functions can, of course, be rapidly formed in real time during a short preceding interval and used as reasonable estimates over the present interval. A less ambitious task is simply to make estimates of short-term waveform-related averages. These include the sample mean (or time-average value), the sample variance (or mean-square value assuming zero mean), and correlation coefficients formed over the previous sample interval. In many applications of waveform analysis, the input waveform is converted to a zero-mean waveform by subtracting the estimates of the mean. This happens, for example, in a digital panel meter in which an auxiliary circuit measures the effects of the internal dc offset voltages and subtracts them in a process known as *auto-zero*. Further, the variance estimate is often used to scale the range of the input waveform to match the dynamic amplitude range of subsequent waveform-handling equipment. This process, performed in the digital panel meter, is called *autoranging* or *automatic gain control* (AGC). The function of these signal conditioning operations, mean removal and variance control (gain adjustment) shown in Figure 11.2, is to normalize the probability density functions of the input waveform. This normalization assures optimal utility of the limited dynamic range of subsequent recording, transmission, or processing subsystems.

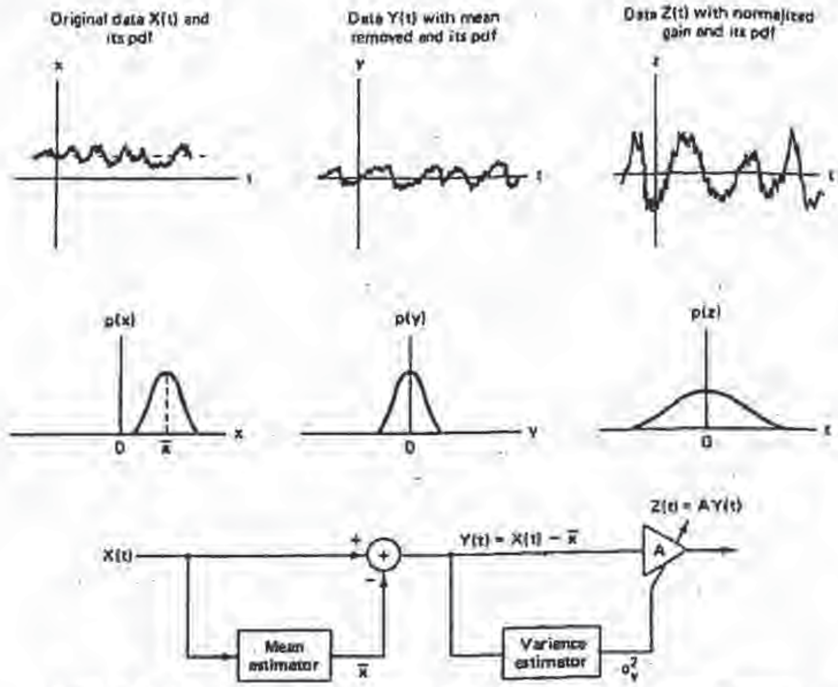


Figure 11.2 Mean removal and variance control (gain adjustment) for a data-dependent signal conditioning system.

11.1.2.2 Autocorrelation Function, Power Spectrum, and Models

There is significant correlation between the amplitudes of many waveform sources in successive time intervals. This correlation means that successive time samples are not independent. If the time sequence is truly independent, the autocorrelation function of the sequence would be an impulse function. The width of the autocorrelation function (in seconds) is called the correlation time of the process and is akin to the time constant of a filter. This time interval is an indication of how much shift along the time axis is necessary to find uncorrelated data samples. If the correlation time is large, we interpret this to mean that the waveform makes significant amplitude changes slowly. Conversely, if the correlation time is small, we infer that the waveform makes significant amplitude changes very quickly.

The Fourier transform of the autocorrelation function is the power spectral density of the waveform process. Thus an alternative description of the autocorrelation function, which reflects the amount of intersample dependence, is the degree of flatness in the waveform power spectrum. A flat spectrum, sometimes called a *white spectrum*, corresponds to source waveforms with independent values sample to sample. A power spectrum with a wide bandwidth implies a time function capable of rapid changes in envelope, while a power spectrum with a narrow bandwidth suggests a time function capable of only slow changes. In general, the larger the deviation from flatness, the more correlation will be found in the waveform samples. Very large changes from flatness in the power spectrum may warrant source descriptions which partition the spectrum, via filters, into subbands each of which is described and quantized separately.

11.2 AMPLITUDE QUANTIZING

Amplitude quantizing is the task of mapping samples of a continuous amplitude waveform to a finite set of amplitudes. The hardware that performs the mapping is the analog-to-digital converter (ADC or A-D). The amplitude quantizing occurs after the sample-and-hold operation. The simplest quantizer to visualize performs an instantaneous mapping from each continuous input sample level to one of the preassigned equally spaced output levels. Quantizers that exhibit equally spaced increments between possible quantized output levels are called *uniform quantizers* or sometimes *linear quantizers*. Possible instantaneous input–output characteristics are easily visualized by a simple staircase graph consisting of risers and treads of the types shown in Figure 11.3. Figure 11.3a, b, and d show quantizers with uniform quantizing steps, while Figure 11.3c is a quantizer with nonuniform quantizing steps. Figure 11.3a depicts a quantizer with *midtread* at the origin, while Figure 11.3b and d present quantizers with *midrisers* at the origin. A distinguishing property of midriser and midtread converters is related to the presence or absence, respectively, of output level changes when the input to the converter is idle noise. Further, Figure 11.3d presents a *biased* (i.e., truncation) quantizer, while the remaining quantizers in the figure are unbiased and are referred to as *rounding quantizers*. Most quantizers are truncation quantizers due to implementation considerations. The terms "midtread" and "midriser" are staircase terms used to describe whether the horizontal or vertical member of the staircase is at the origin. The unity-slope line passing through the origin represents the ideal nonquantized input–output characteristic we are trying to approximate with the staircase. The difference between the staircase and the unity-slope-line segment represents the approximation error made by the quantizer at each input level. Figure 11.4 illustrates the approximation error amplitude versus input amplitude function for each quantizer characteristic in Figure 11.3. Parts (a) through (d) of Figure 11.4 correspond to the same parts in Figure 11.3. This error is often modeled as quantizing noise because the error sequence obtained when quantizing a

DIGITAL CODING OF WAVEFORMS
Principles and Applications to Speech and Video

N. S. JAYANT
Bell Laboratories, Inc.
Murray Hill, NJ

PETER NOLL
Technical University of Berlin

ing

PRENTICE-HALL, INC. Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging in Publication Data

Jayant, Nuggehally S., 1946-
Digital coding of waveforms.

Includes index.
1. Signal processing—Digital techniques. 2. Coding theory. I. Noll, P. (Peter), 1936- II. Title.
TK5102.J39 1984 621.38'043 83-22170
ISBN 0-13-211913-7

Editorial/production supervision: *Shari Ingerman*
Cover design: *Edsal Enterprises*
Manufacturing buyer: *Tony Caruso*
Page layout: *Diane Koromhas*

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-211913-7 01

Prentice-Hall International, Inc., *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*
Whitehall Books Limited, *Wellington, New Zealand*

11

Sub-Band Coding

11.1 Introduction

In the class of *time domain* coding algorithms (Chapters 5 to 10), the input waveform is treated as a single full-band signal; and in predictive coders, redundancy is removed prior to encoding by prediction and inverse filtering. The main differences in the various algorithms are determined by the degree of prediction (Chapters 6 to 8) that is employed, and by whether schemes are adaptive or not. In delayed decision coding (Chapter 9), input structure is exploited by means of a multipath search.

In Chapters 11 and 12 another class of encoding algorithms will be discussed in which the approach is to divide the input signal into a number of separate frequency components, and to encode each of these components separately. This division into frequency components removes the redundancy in the input and provides a set of uncorrelated inputs to the channel. Recall that the action of a DPCM coder is also similar, if not identical. The encoder in that case, when fed by a redundant signal, outputs a sequence of prediction error components that tend to be uncorrelated. The *frequency domain* coding techniques have the advantage that the number of bits used to encode each frequency component can be variable, so that the encoding accuracy is always placed where it is needed in the frequency domain. In fact, bands with little or no energy may not be encoded at all. Variable bit allocation can in principle provide arbitrary forms of noise shaping, a feature that was realized to some extent by noise feedback in the time-domain methods of Chapter 7.

As in the case of time domain techniques, a large variety of frequency domain algorithms, from simple to complex, are available and the main differences are usually determined by the way in which source statistics are modeled, and the degree to which source redundancy is exploited, in the technique. We will begin by describing one technique of lower complexity called ***Sub-Band Coding*** (SBC) and then proceed to one of higher complexity called ***Transform Coding*** (TC) (Chapter 12). In the notation of Chapter 1, SBC with fixed bit allocation will be a *medium-complexity* coder and TC with variable bit allocation will be a ***high-complexity*** coder.

Unless otherwise mentioned, focus in this SBC chapter will be on speech waveforms. In the sub-band coder the speech band is divided into typically four or more sub-bands by a bank of bandpass filters. Each sub-band is, in effect, lowpass translated to zero frequency by a modulation process equivalent to single-side-band amplitude modulation. It is then sampled (or resampled) at its Nyquist rate (twice the width of the band) and digitally encoded with a PCM or DPCM encoder [Crochiere, Webber and Flanagan, 1976] [Esteban and Galand, 1978]. In this process, each sub-band can be encoded according to perceptual criteria that are specific to that band. On reconstruction, the sub-band signals are decoded and modulated back to their original locations. They are then summed to give a close replica of the original speech signal.

Encoding in sub-bands offers several advantages. By appropriately allocating the bits in different bands, the number of quantizer levels and hence reconstruction error variance can be separately controlled in each band, and the shape of the overall reconstruction error spectrum can be controlled as a function of frequency. In the lower frequency bands, where pitch and formant structure must be accurately preserved, a larger number of bits/sample can be used; whereas in upper frequency bands, where fricative and noise-like sounds occur in speech, fewer bits/sample can be used. Further, quantization noise can be contained within bands to prevent masking of a low-level input in one frequency range by quantizing noise in another frequency range. Section 11.2 gives a quantitative demonstration of objective (SNR) gains due to sub-band coding.

The most complex part of the coder is the filter bank [Bellanger, Bonnerot and Coudreuse, 1976] [Esteban and Galand, 1977]. With newer filter technologies such as CCD filters and digital filters, this complexity is rapidly being reduced. Also the design technique of *quadrature-mirror filters* (Section 11.4) affords distinct advantages in digital implementation of this coder.

Figure 11.1 illustrates a basic block diagram of the sub-band coder. The coder consists of a bank of M bandpass filters, followed by sub-band encoders which typically are PCM-AQB coders, and a multiplexer. The receiver has the inverse stages of demultiplexing, decoding and bandpass filtering prior to sub-band addition. Unlike the spectrum channel vocoder for synthetic speech [Flanagan et al., 1979] [Rabiner and Schafer, 1978] where the object of the filter-bank is only to preserve information about short-time energy as a function of frequency, the sub-band coder in Figure 11.1 transmits individual time waveforms $x_k(t)$; $k = 1,2,..., M$ and the receiver adds decoder versions $y_k(t)$ phase-synchronously to obtain $y(t)$. The sub-band coder is therefore a waveform-preserving coder.

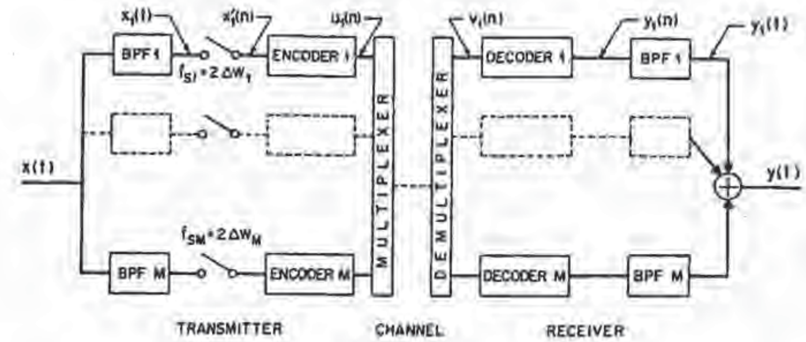


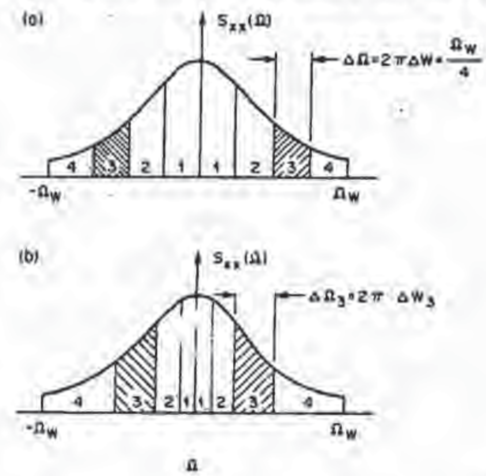
Figure 11.1 Block diagram of sub-band coding (SBC).

In Figure 11.1, sub-band width ΔW_k was a function of sub-band number k, implying *variable-width* sub-bands. The special case of *equal-width* sub-bands is important for implementation (Section 11.4) as well as analytical tractability (Section 11.2). Both types of arrangements will be considered for the sub-band coding of speech (Section 11.5). Figure 11.2 illustrates the two classes of arrangements for the example of $M = 4$. Shaded regions define sub-band number $k = 3$.

In the case of *equal-width* sub-bands

$$\Delta W_k = \Delta W = W/M \; ; \quad k = 1,2,...,M$$
$$\Delta\Omega = 2\pi\Delta W = \Omega_W/M = 2\pi W/M \tag{11.1}$$

Figure 11.2 Division of input spectrum into $M = 4$ sub-bands of (a) constant and (b) variable width.



where W and Ω_W represent the total input bandwidth in Hz and radians/second, respectively. In the case of unequal sub-bands, they are typically made wider as k increases:

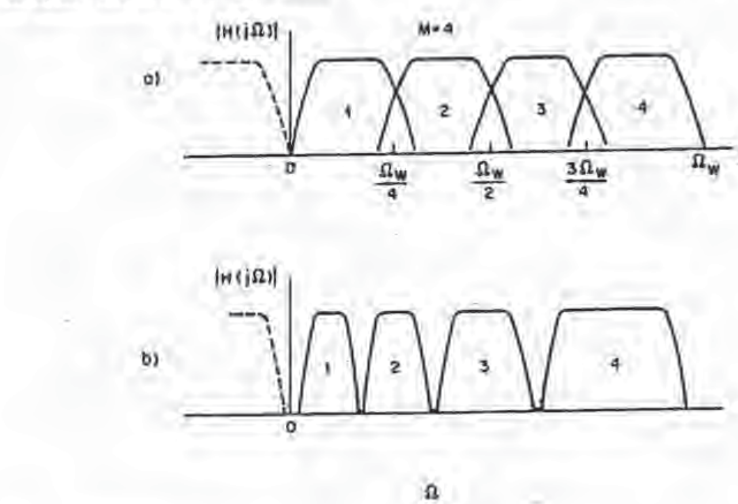
$$\Delta W_{k+1} > \Delta W_k \ ; \ k = 1,2,...,M-1 \qquad (11.2)$$

The design in (11.2) suggests that the lower frequencies in the speech signal are more carefully isolated or observed than the higher frequencies. This provides a qualitative match to the long-time speech psd [Figure 2.9(a)] and the articulation-index function (Appendix E), both of which are lowpass functions that decrease as frequency increases. The above match is, however, not very critical in the presence of variable bit allocation, which offers the possibility of digitizing sub-bands with varying fidelity (Section 11.3). Indeed, an important filter-bank design (the quadrature-mirror filter bank, Section 11.4) has the defining characteristic that the input psd is split into equal sub-bands.

Figure 11.3 sketches filter-bank amplitude responses that may be appropriate to realize the band-splitting operations shown in Figure 11.2. The observations to be made with Figure 11.3 will also hold for the digital filter banks of Section 11.3, with $H(j\Omega)$ and Ω_W replaced by $H(e^{j\omega})$ and π, respectively. An important distinction in Figure 11.3 is between (a) equal-width and (b) variable-width filters. Another distinction is between filter characteristics that overlap, as in (a), and characteristics that are non-contiguous, as in (b). The in-between situation of exactly contiguous filters is academic because practical implementations involve amplitude responses with finite roll-off characteristics.

The approach in Figure 11.3(b) calls for extremely fast filter roll-offs that minimize inter-band gaps, but it offers the possibility of reduced sampling rates

Figure 11.3 Amplitude responses in filter-banks consisting of four individual bandpass characteristics of (a) equal width and (b) unequal width.



(smaller values of f_{sk}), and hence a lower coding rate I [see (11.3)] for given values of R_k. Inter-band frequency gaps will be non-zero in practical filter designs, and these gaps cause a reverberant quality in the output speech of low bit rate SBC designs, unless the sub-bands can adaptively track regions of significant speech energy, such as formant frequencies in voiced speech [Crochiere and Sambur, 1977]. Discussions of SBC in this book are confined to fixed bands and fixed bit allocations. Sub-band coding systems with adaptive bit allocation perform significantly better because of the dynamic noise-shaping that they provide [Esteban and Galand, 1978] [Grauel, 1980] [Ramstad, 1982] [Heron, Crochiere and Cox, 1983]. However, the higher processing involved in adaptive bit allocation makes it particularly appropriate in the higher-complexity approach of TC (Chapter 12).

11.2 Transmission Rate *I*, *SNR* and Gain Over PCM

In SBC, each sub-band waveform $x_k(t)$ is sampled at a rate f_{sk} and encoded using R_k bits per sub-band sample. The transmission rate in SBC is therefore the sum of the bit rates needed to code individual sub-bands:

$$I = \sum_{k=1}^{M} f_{sk} R_k \quad \text{b/s} \tag{11.3}$$

In the special case of *equal-width* sub-bands,

$$\Delta W_k = W/M \quad \text{for all } k \; ; \quad f_{sk} = 2\Delta W_k = 2W/M \tag{11.4a}$$

Since individual sub-band k can be sampled at the frequency $2\Delta W_k$ (Section 11.3), (11.3) simplifies to

$$I = \frac{2W}{M} \sum_{k=1}^{M} R_k \quad \text{b/s} \qquad \text{for equal-width bands} \tag{11.4b}$$

Note that (11.4b) reduces to the familiar form $I = 2WR$ for full-band coding if the total number of bits is expressed in the form

$$\sum_{k=1}^{M} R_k = MR \tag{11.5}$$

where R denotes the average number of bits used to encode a full-band sample. The simple equalities in (11.4b) and (11.5) imply that I is proportional to the sum of R_k values. This makes the design of variable bit allocation much simpler than in the general case of unequal-width sub-bands where the relationship between total bit rate I and the individual R_k values is less direct [see (11.3)].

In the following analysis, we assume non-overlapping equal-width sub-bands, so that the variances σ_{xk}^2 of sub-band inputs can be simply added to obtain the variance σ_x^2 of the full-band input. Similarly, variances σ_{rk}^2 of sub-band

1.3)] for given
il filter designs,
w bit rate SBC
nificant speech
and Sambur,
s and fixed bit
ation perform
they provide
ron, Crochiere
: bit allocation
roach of TC

ĊM

and encoded
therefore the

(11.3)

(11.4a)

Section 11.3),

(11.4b)

nd coding if

(11.5)

and sample.
l to the sum
pler than in
etween total

ib-bands, so
obtain the
f sub-band

reconstruction errors can be added to obtain the variance σ_r^2 of signal reconstruction error.

The final reconstruction error variance is

$$\sigma^2_{r,SBC} = \sum_{k=1}^{M} \sigma^2_{rk} \tag{11.6a}$$

We assume error-free transmission, and the use of PCM (or DPCM) coding of individual sub-bands. As a result, for any k, the sub-band reconstruction error variance is $\sigma^2_{rk} = \sigma^2_{qk}$, a corresponding quantization error variance. Therefore,

$$\sigma^2_{r,SBC} = \sum_{k=1}^{M} \epsilon^2_{*k}\, 2^{-2R_k}\, \sigma^2_{xk} \tag{11.6b}$$

The reconstruction error variance of a conventional (full-band) PCM coder, with bit rate equal to the average bit rate R in (11.5), is given by

$$\sigma^2_{r,PCM} = \epsilon^2_* 2^{-2R} \sigma^2_x \tag{11.6c}$$

where R is the number of bits/sample.

The SNR improvement G_{SBC} due to sub-band coding is the ratio of (11.6c) to (11.6b). Assuming for simplicity a *constant quantizer performance factor* ($\epsilon^2_{*k} = \epsilon^2_*$; all k), we obtain a gain over PCM that depends *only* on the bit allocation algorithm:

$$G_{SBC} = \frac{2^{-2R}\,\sigma^2_x}{\sum_{k=1}^{M}\left[2^{-2R_k}\sigma^2_{xk}\right]} = \frac{2^{-2\sum R_k/M}\sum_{k=1}^{M}\sigma^2_{xk}}{\sum_{k=1}^{M}\left[2^{-2R_k}\sigma^2_{xk}\right]} \tag{11.7a}$$

$$SNR\,|_{SBC}(\text{dB}) = SNR\,|_{PCM}(\text{dB}) + 10 \log G_{SBC} \tag{11.7b}$$

With a flat spectrum, G_{SBC} can never exceed 1 (Example 11.1). In the case of non-flat spectra, values of $G_{SBC} > 1$ can be realized by *bit allocation* procedures where R_k values are matched to σ^2_{xk} values in a sense that will be clear from Examples 11.1 and 11.2. The special case of σ^2_{xk}-independent and equal R_k simply leads to $G_{SBC} = 1$ in (11.7). In Chapter 12, we will fully develop a theory of optimum bit allocation.

The sub-band coding gain G_{SBC} is really analogous to the prediction gain G_P in that both of these gains result from the non-flatness of input spectrum. Sub-band coding gain increases as a function of number of bands M; and prediction gain increases with order of prediction. As in prediction, the greatest values of G_{SBC} are realized when spectrum-dependent bit allocation is allowed to be time varying [Stjernvall, 1977] [Esteban and Galand, 1978] [Grauel, 1980] [Ramstad, 1982] [Heron, Crochiere and Cox, 1983]. This will indeed be the approach of Adaptive Transform Coding in Chapter 12.

Example 11.1. Two-band coding of a flat spectrum input

We shall again consider the simple case of equally wide sub-bands of width $W/2$ each. As a result of the flat spectrum,

$$\sigma_{x1}^2 = \sigma_{x2}^2 = \sigma_x^2/2 \tag{11.8a}$$

Using this in (11.7a),

$$G_{SBC} = 2\,\frac{2^{-(R_1+R_2)}}{2^{-2R_1} + 2^{-2R_2}} \tag{11.8b}$$

Figure 11.4(a) plots G_{SBC} as a function of R_1 for the example of $R_1 + R_2 = 6$. The maximum value of $G_{SBC} = 1$ occurs for $R_1 = R_2 = 3$. This result can also be seen by identifying the above expression for G_{SBC} as the ratio of geometric mean and arithmetic mean of the terms 2^{-2R_1} and 2^{-2R_2}; this ratio is maximum when the terms are equal, i.e., when $R_1 = R_2$. ■

Example 11.2. Two-band coding of the two-level spectrum of Figure 2.24 with $\alpha = 2/17$

From the results of Example 2.12,

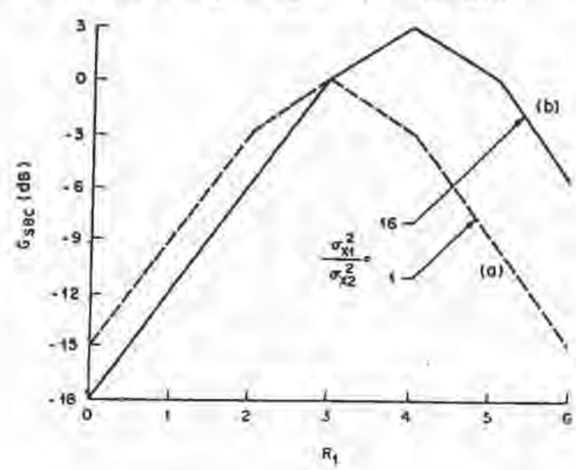
$$\sigma_{x1}^2 = (16/17)\,\sigma_x^2\,; \quad \sigma_{x2}^2 = (1/17)\,\sigma_x^2 \tag{11.9a}$$

Using this in (11.7a),

$$G_{SBC} = 17\,\frac{2^{-(R_1+R_2)}}{16\cdot 2^{-2R_1} + 1\cdot 2^{-2R_2}} \tag{11.9b}$$

Figure 11.4(b) plots G_{SBC} as a function of R_1 for the example of $2R = R_1 + R_2 = 6$. As in Example 11.1, equal bit allocation ($R_1 = R_2 = 3$) results in $G_{SBC} = 1$. But unlike in the flat-spectrum case, the maximum value of G_{SBC} is now 17/8. This maximum occurs if

Figure 11.4 G_{SBC} versus R_1 in two-band SBC schemes with $R = 3$ bits/sample, for (a) a flat-spectrum input, and (b) an input with the two-level psd of Figure 2.24 (with $\alpha = 2/17$).



DISH-Blue Spike-246
Exhibit 1010, Page 0457

$R_1 = 4$ and $R_2 = 2$. Note that with this design, the error contributions in both sub-bands (denominator terms in the above expression for G_{SBC}) are equal. In fact, this maximum gain is the inverse of the spectral flatness measure γ_x^2 for the input (Problem 2.23):

$$\gamma_x^2 = [\alpha(2-\alpha)]^{1/2} = \left[\frac{2}{17}\,\frac{32}{17}\right]^{1/2} = \frac{8}{17}$$

The equality of maximum gain and γ_x^{-2} will be discussed again in Chapter 12. What is significant for the present discussion is that a two-band SBC procedure is sufficient to utilize all the spectral redundancy of a two-level psd: in other words, sufficient to realize $G_{SBC} = \gamma_x^{-2}$. The exact realization of the maximum possible gain is due to one other property of the input spectrum in question: the ratio of component variances, 16, is an integral power of 2.

The sufficiency of the two-band partition for the two-level spectrum is similar to a result in Chapter 6 where linear prediction of order N was adequate to utilize the *sfm* of an AR(N) process. ■

The constraint on $\sum R_k$ used in Examples 11.1 and 11.2 is more meaningful with equal-width sub-bands [where I is directly proportional to $\sum R_k$ (11.4b)] than with variable-width sub-bands [where I is a weighted sum of the R_k (11.3)]. For equal-width sub-bands, the results of Example 11.2 can be generalized to the case of $M > 2$. The gain G_{SBC} in (11.7a) can again be maximized under the constraint of a given number of bits [see (11.5)]. This is equivalent to minimizing

$$\sigma_r^2 = \epsilon_*^2 \sum_{k=1}^{M} 2^{-2R_k}\,\sigma_{xk}^2$$

Using Lagrange multipliers,

$$\frac{\partial}{\partial R_k}\left[\epsilon_*^2 \sum_{k=1}^{M} 2^{-2R_k}\,\sigma_{xk}^2 - \lambda\left[MR - \sum_{k=1}^{M} R_k\right]\right] = 0$$

from which we can express R_k as a function of λ:

$$R_k = \frac{1}{2}\log_2\left[2\epsilon_*^2\log_e 2\right] + \frac{1}{2}\log_2\frac{\sigma_{xk}^2}{\lambda}$$

Using this result in $MR = \sum R_k$, the *optimum bit allocation* is

$$R_{k,opt} = R + \frac{1}{2}\log_2\frac{\sigma_{xk}^2}{\left[\prod_{l=1}^{M}\sigma_{xl}^2\right]^{1/M}} \tag{11.10}$$

and from the expression for σ_r^2 above, the *minimum mse* [Goodman, 1967] is

$$\min\{\sigma_r^2\} = M\,\epsilon_*^2 2^{-2R}\left[\prod_{l=1}^{M}\sigma_{xl}^2\right]^{1/M} \tag{11.11}$$

and the *maximum gain* is the ratio of arithmetic mean and geometric mean of the sub-band variances σ_{xl}^2:

$$\max(G_{SBC}) = \frac{\sigma_x^2}{M\left[\prod_{l=1}^{M}\sigma_{xl}^2\right]^{1/M}} = \frac{\frac{1}{M}\sum_{l=1}^{M}\sigma_{xl}^2}{\left[\prod_{l=1}^{M}\sigma_{xl}^2\right]^{1/M}} \tag{11.12}$$

The bit allocation in (11.10), which minimizes the mse σ_r^2, will also imply equal values of noise variance for different k, as a result of (11.6b) and (11.10); this is also shown formally in the analysis of Chapter 12. We will also see in Chapter 12 that specific forms of noise-shaping can be realized by bit allocations that minimize certain types of frequency-weighted mean square error.

In coding a signal such as speech whose psd can be approximated as an M-level psd with $M >> 2$ (a generalization of Figure 2.24), SBC performance increases with increasing M. Very large values of M can also take into account the fine structure in the psd due to the pitch period, in the sense of maintaining a locally flat psd within each sub-band. The use of fairly small values such as $M = 4$ is therefore for simplicity, rather than because of a saturation of objective gain G_{SBC} as a function of M. In particular, 4-band SBC coding of speech is significantly better than 2-band SBC coding, both objectively and from a subjective quality viewpoint [Cox, 1981, II]. Improvement of performance with M is maintained at values as high as $M = 16$ [Ramstad, 1982] [Esteban and Galand, 1982].

11.3 The Integer-Band Filter Bank

An important feature in Figure 11.1 is that bandpass filter cutoffs are chosen such that each band can be sampled at twice the corresponding bandwidth

$$f_{sk} = 2\Delta W_k\ ; \qquad k = 1,2,\ldots,M \tag{11.13}$$

rather than at twice the highest frequency of the full-band signal. As discussed in Chapter 3, this is possible in the special situation of *integer-band sampling* [Crochiere and Rabiner, 1983] where the lower cutoff frequency W_{lk} in a sub-band k is an integral multiple of bandwidth (Figure 3.10):

$$W_{lk} = n\,\Delta W_k\ ; \quad n = 0,1,2,\ldots; \quad k = 1,2,3,\ldots,M \tag{11.14}$$

The orders of bandpass filtering and sampling in Figure 11.1 can be reversed. Consider that discrete-time inputs $x(n)$ are available, sampled at the full-band Nyquist rate $f_s = 2W$ where $W = \sum \Delta W_k$, with summation from $k = 1$ to M, is the maximum frequency of the full-band signal.

Let the sub-band width be written in the form

$$\Delta W_k = W/\zeta_k\ ; \qquad k = 1,2,\ldots,M \tag{11.15}$$
$$\zeta_k = M \text{ for all } k \text{ with } \textit{equal-width} \text{ sub-bands}$$

DISH-Blue Spike-246
Exhibit 1010, Page 0459

tric mean of the

(11.12)

lso imply equal
(11.10); this is
e in Chapter 12
s that minimize

d as an M-level
nance increases
ccount the fine
aining a locally
h as $M = 4$ is
ctive gain G_{SBC}
is significantly
jective quality
maintained at
82].

offs are chosen
width

(11.13)

As discussed in
and sampling
in a sub-band

(11.14)

n be reversed.
the full-band
$= 1$ to M, is

(11.15)

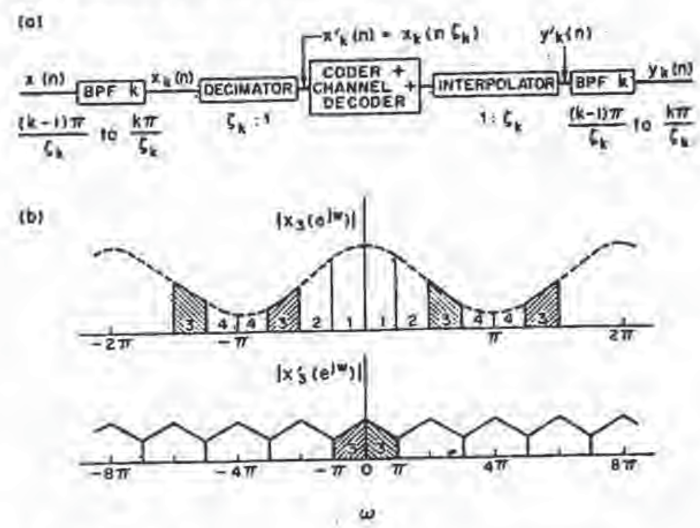
With the sub-band partition of Figures 11.2(a) and 11.3(a), $\zeta_k = 4$ for all k. Some of the speech coding examples mentioned later in this chapter use unequal sub-bands. In these examples, the values of ζ_k range from 4 to 30 (Tables 11.1, 11.2 and 11.3).

Figure 11.5 shows the sequence of filtering and coding operations in SBC, using the general example of sub-band k in Figure 11.5(a) and the special case of $k = 3$ and $\zeta_3 = 4$ in Figure 11.5(b).

The ζ_k:1 *decimator* sub-samples the bandpass output $x_k(n)$ by a factor ζ_k, implying a sampling rate of $f_{sk} = 2W/\zeta_k = 2\Delta W_k$ for sub-band k. This decimation implies a repetition rate of its spectrum that is higher than that of the full-band spectrum by a factor ζ_k. As a result of this, the x-axes in the two figures of (b) differ by a factor $\zeta_k = 4$. One of the repetitions of the spectrum will be in the baseband, so that the decimation effectively translates the lower frequency edge of the bandpass signal band to zero frequency. The 1:ζ_k *interpolator* fills in $(\zeta_k - 1)$ zeros in between every pair of incoming lowpass samples. The kth harmonic of the interpolated baseband is thus effectively bandpass-translated to the appropriate initial bandpass region. The explicit modulation processes mentioned in Section 11.1 are therefore replaced by simpler discrete-time processes of decimation and interpolation. It is assumed that the interpolation process includes an amplitude scaling factor of ζ_k. This maintains the original value of input variance in spite of the zero-valued amplitudes that are introduced in the interpolation process.

The amplitude spectra $|X_3(e^{j\omega})|$ and $|X'_3(e^{j\omega})|$ in Figure 11.5(b) refer to sub-band $k = 3$, with $\zeta_3 = 4$; the illustration is equivalent to the continuous-time case

Figure 11.5 Realization of integer-band sampling with a discrete-time input: (a) block diagram of SBC coding for sub-band k; and (b) original spectrum and resampled spectrum after decimation, for sub-band $k = 3$, with $\zeta_3 = 4$. The baseband spectrum resulting from the decimation is shifted back to the original frequency range of sub-band k after interpolation by a factor ζ_k.



DISH-Blue Spike-246
Exhibit 1010, Page 0460

of Figure 3.9, which also used the example of $k = 3$. Note that the spectrum of the decimated sequence has its own frequency scaling. If the procedure of the last two paragraphs is repeated for an even-numbered sub-band ($k = 2$ or $k = 4$), it can be shown that the spectrum gets inverted in the process of lowpass translation to the baseband. This is, however, neutralized by a subsequent inversion in the interpolation process for even k [Crochiere and Rabiner, 1983].

The integer-band constraint in (11.14) is invariably assumed in SBC for the obvious reason of minimizing sub-band sampling frequencies and hence the overall information rate

$$I = \sum_{k=1}^{M} I_k = \sum_{k=1}^{M} f_{sk} R_k = \sum_{k=1}^{M} 2\Delta W_k \, R_k \quad \text{bits/second} \qquad (11.16)$$

11.4 Quadrature-Mirror Filter Banks

The overlapping sub-band situation in Figure 11.3(a) suggests that aliasing effects can occur if sub-bands are sampled at $f_{sk} = 2W/M = \Omega_W/\pi M$. This problem is very elegantly tackled in the *quadrature-mirror filter bank* (QMFB) approach of Figure 11.6 [Esteban and Galand, 1977]. This figure shows the division of a full-band signal of maximum radian frequency π into two of equal width by using a constrained pair of lowpass and highpass filters. In the notation of (11.15), $\zeta_1 = \zeta_2 = M = 2$. By repeated subdivisions of resulting sub-bands using QMF filter banks, one can realize an SBC filter bank with M given by a power of 2, such as $M = 4$ as in Figure 11.3(a). Values of M that are not powers of 2 can also be realized by simply ignoring appropriate sub-band branches in the QMF tree (Table 11.3 and Problem 11.2).

The rest of the following discussion refers to the first stage of such a filter bank, involving two sub-bands as in Figure 11.6. When further stages of band-partitioning are introduced, each of the branches in Figure 11.6(a) will be split into further branches, and sampling frequencies will be reduced by factors of two at each stage; but the results of Figure 11.6(b) will apply repeatedly with appropriate redefinitions of the absolute frequencies represented by 0, π and $\pi/2$ in that figure.

Each of the sub-band signals $x_l(n)$ and $x_u(n)$ is resampled by a factor 2:1. This reduction of the sub-band sampling rates is necessary in order to maintain a minimal overall bit rate in encoding these signals. This reduction of sampling rate introduces aliasing terms in each of the sub-band signals because of the finite rate of roll-off in filter responses. For example, in the lower band the signal energy in the frequency range above $\pi/2$ is folded down into the range 0 to $\pi/2$ and appears as *aliasing distortion* in this signal, in the frequency range covered by the hatched region in the left half of Figure 11.6(b). In the above explanation, and in the rest of this section, π is *not* redefined as in Figure 11.5(b). Aliasing also occurs for the upper band in a similar fashion; any signal energy in the frequency range below $\pi/2$ is folded upward into its Nyquist band $\pi/2$ to π; this causes aliasing in the frequency range covered by the hatched area in the right half of Figure 11.6(b). This mutual aliasing of signal energy between the upper and lower sub-bands is

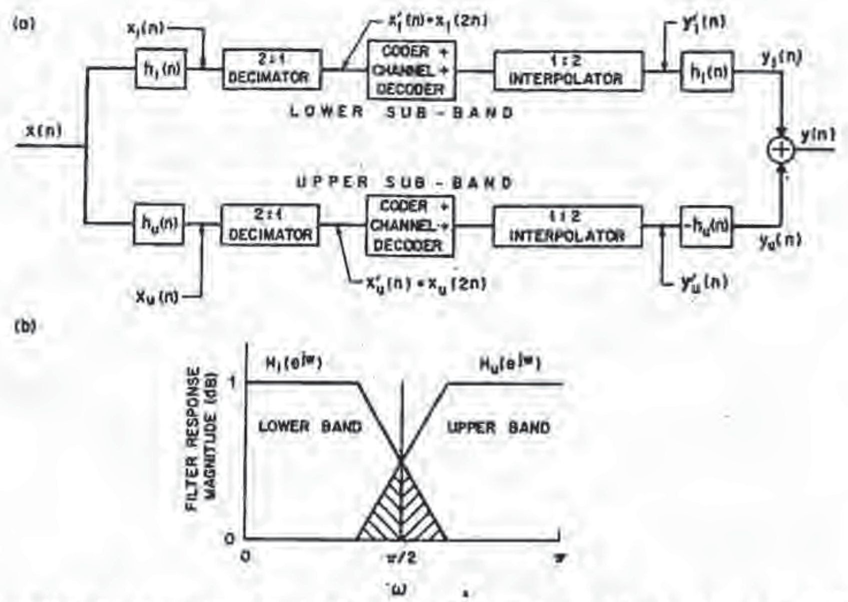


Figure 11.6 Quadrature-mirror filtering for splitting an input into two equal-width sub-bands: (a) implementation; and (b) qualitative illustration of a filter-bank response that provides alias-image cancellation [Esteban and Galand, 1978] [Crochiere, 1981].

sometimes called *interband leakage*. The amount of leakage that occurs between sub-bands is directly dependent on the degree to which the filters $h_l(n)$ and $h_u(n)$ approximate ideal lowpass and highpass filters, respectively.

In the reconstruction process, the sub-band sampling rates are increased by a factor 1:2 by filling in zero-valued samples between each pair of sub-band samples. This introduces a periodic repetition of the signal spectra in the sub-band. For example, in the lower band the signal energy from 0 to $\pi/2$ is symmetrically folded around the frequency $\pi/2$ into the range of the upper band. This unwanted signal energy, referred to as an *image* is mostly filtered out by the lowpass filter $h_l(n)$ in the receiver. This filtering operation effectively interpolates the zero-valued samples that have been inserted between the sub-band signals to values that appropriately represent the desired waveform [Crochiere and Rabiner, 1983]. Similarly, in the upper sub-band signal an image is reflected to the lower sub-band and filtered out by the filter $-h_u(n)$.

The degree to which the above images are removed by the filters $h_l(n)$ and $-h_u(n)$ is determined by the degree to which they approximate ideal lowpass and highpass filters. Because of the special relationship of the sub-band signals in the QMF filter bank, the remaining components of the images can be canceled by aliasing terms introduced in the analysis. This cancellation occurs *after* the addition of the two interpolated sub-band signals $y_l(n)$ and $y_u(n)$, and the cancellation is exact in the absence of coding errors. In the presence of coding, this cancellation is obtained to the level of quantization noise.

If S_l and S_u refer to lower-band and upper-band signals and subscripts A and I denote aliasing and imaging, the adder input $y_l(n)$ in Figure 11.6(a) will consist of the following main components, correct to filter attenuation effects: S_l and S_{uA} in the lower band and S_{lI} and S_{luA} in the upper band. Similarly, input $y_u(n)$ will consist of components S_{uI} and S_{lIA} in the lower band and S_u and S_{lA} in the upper band. When $y_l(n)$ and $y_u(n)$ are added, components S_{uA} and S_{uI} cancel in the lower band, while components S_{lI} and S_{lA} cancel in the upper band, leaving S_l and S_{lIA} in the lower band and S_u and S_{luA} in the higher band. The components of S_{lIA} and S_{luA} actually belong in the respective bands, and in the case of a QMF design with an allpass characteristic, these components exactly compensate for the in-band attenuations present in S_l and S_u.

One way of obtaining this cancellation property in the QMF filter bank is to use filters $h_l(n)$ and $h_u(n)$, which are respectively symmetrical and anti-symmetrical *finite impulse response* (FIR) designs with even numbers of taps, i.e.,

$$h_l(n) = h_u(n) = 0 \quad \text{for } 0 > n \geqslant N \ ; \tag{11.17}$$

$$h_l(n) = h_l(N-1-n) \ , \quad n = 0,1,...,N/2-1 \ ; \tag{11.18a}$$

$$h_u(n) = -h_u(N-1-n) \ , \quad n = 0,1,...,N/2-1 \tag{11.18b}$$

The cancellation of aliasing effects in the QMF filter bank further requires that the filters in Figure 11.6(a) satisfy the condition [Esteban and Galand, 1977] [Crochiere and Rabiner, 1983]

$$h_u(n) = (-1)^n h_l(n) \ , \quad n = 0,1,...,N-1 \tag{11.19}$$

which is a *mirror image* relationship of the filters, implying symmetry about $\pi/2$, as in Figure 11.6(b). The coefficients of the filters are identical except that their signs alternate. Therefore, both filters can be realized using a single N-tap filter as the starting point.

Further, if the filter-bank output $y(n)$ is desired to be a delayed replica of input $x(n)$ (in the absence of coding errors), the filters $h_l(n)$ and $h_u(n)$ must also satisfy the condition

$$|H_l(e^{j\omega})|^2 + |H_u(e^{j\omega})|^2 = 1 \ , \tag{11.20}$$

where $H_l(e^{j\omega})$ and $H_u(e^{j\omega})$ are the Fourier transforms of $h_l(n)$ and $h_u(n)$, respectively; this is simply the condition for an *allpass* characteristic. If the allpass condition (11.20) is included in the mirror-image design (11.19), the point of intersection of the two filter functions in Figure 11.6(b) will be the -3 dB point for each transfer function.

The filter requirement in (11.20) cannot be met exactly by the mirror image filters of (11.19) except when $N = 2$ and when N approaches infinity. However, it can be very closely approximated for modest values of N. Filter designs which satisfy (11.18) and (11.19) and approximate the condition of (11.20) can be obtained with the aid of an optimization program [Johnston, 1980]. Resulting filter

ubscripts A and I
(a) will consist of
is: S_l and S_{uA} in
input $y_u(n)$ will
S_{lA} in the upper
S_{ul} cancel in the
d, leaving S_l and
ie components of
case of a QMF
mpensate for the

er bank is to use
anti-symmetrical
e.,

(11.17)

(11.18a)

(11.18b)

equires that the
Galand, 1977]

(11.19)

y about $\pi/2$, as
that their signs
ap filter as the

replica of input
ust also satisfy

(11.20)

) and $h_u(n)$,
If the allpass
the point of
3 dB point for

mirror image
. However, it
lesigns which
1.20) can be
esulting filter

Table 11.1 Quadrature Mirror Filters of order $N = 32$ and $N = 16$. Listed numbers are values of coefficients $h_l(n)$ for $N/2 \leqslant n \leqslant N$. Values of other coefficients follow (11.8) and (11.9) [Johnston, 1980] [Crochiere and Rabiner, 1983; Reprinted with permission].

$N = 32$ $h(16)$ to $h(23)$	$h(24)$ to $h(31)$	$N = 16$ $h(8)$ to $h(15)$
4.6645830E-01	1.7881950E-02	.47211220E 00
1.2846510E-01	−1.7219030E-04	.11786660E 00
−9.9800110E-02	−9.3636330E-03	−.99295500E-01
−3.9244910E-02	1.4272050E-03	−.26275600E-01
5.2909300E-02	4.1581240E-03	.46476840E-01
1.4468810E-02	−1.2601150E-03	.19911500E-02
−3.1155320E-02	−1.3508480E-03	−.20487510E-01
−4.1094160E-03	6.5064660E-04	.65256660E-02

coefficients for N values in the range 8 to 64 have been tabulated [Crochiere, 1981] [Crochiere and Rabiner, 1983].

Table 11.1 lists representative designs for $N = 32$ and $N = 16$. In a QMF tree for $M = 4$ sub-bands, the first subdivision may use filters of order $N = 32$. In view of the 2:1 decimation, a matching design for the second stage of the QMF tree would then be $N = 16$.

Figure 11.7 shows the frequency response characteristics for a 32-tap filter design [Crochiere, 1981]. Figure 11.7(a) shows the magnitude of $H_l(e^{j\omega})$ and $H_u(e^{j\omega})$ expressed in dB as a function of ω. As in the schematic of Figure 11.6(b), note that the roll-off regions of the two responses intersect at the −3dB point for each characteristic. Figure 11.7(b) shows the magnitude of the expression $|H_l(e^{j\omega})|^2 + |H_u(e^{j\omega})|^2$ expressed in dB as a function of ω. As can be seen from Figure 11.7(b), the requirement of (11.20) is satisfied to within ±0.025 dB, which is more than satisfactory for good SBC performance. The reconstruction error in the 32-tap design of Table 11.1 is also ±0.025 dB, but the stop-band attenuation of this filter (measured at the first stop-band peak) is 52 dB, which is much greater than the 37 dB attenuation in the example of Figure 11.7(a). In the 16-tap example of Table 11.1, the reconstruction error is ±0.07 dB, and the stop-band attenuation is 30 dB.

The use of FIR filters has the advantage of linear-phase characteristics which eliminate the problems of group delay distortions. This feature also allows the 2-band design of Figure 11.6 to be conveniently cascaded in three structures (Example 11.5, Problem 11.2) without the need for phase compensation. However, effective FIR designs imply significant coding delays. For example, with the 32-tap design just mentioned, the coding and decoding delays due to the first level of the QMF tree are 4 ms each, assuming 8 kHz input sampling; and subsequent levels of the QMF partition introduce corresponding additional delays. In order to implement SBC systems with smaller values of delay, there has been at least one proposal for a QMF bank based on *infinite impulse response* (IIR) designs [Ramstad and Foss, 1980]. This proposal includes special procedures for mitigating the group delay distortions inherent in IIR designs.

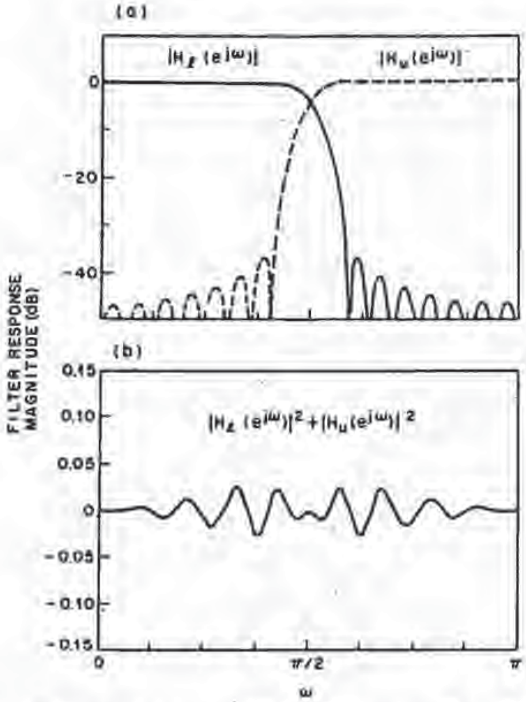


Figure 11.7 Illustration of amplitude-frequency responses in a quadrature mirror filter bank using FIR filters: (a) lowpass and highpass characteristics of individual 32-tap FIR filters; and (b) approximately allpass characteristic of the combination [Crochiere, 1981].

11.5 Sub-Band Coding of Speech

The following examples illustrate the design of fixed bit-allocation SBC systems for speech at bit rates in the range of 9.6 kb/s to 32 kb/s. The 32 kb/s system with fixed bit allocation can provide very high subjective quality, with MOS scores in the order of 4.2, a necessary condition for ***toll quality*** reproduction of telephone speech, while fixed bit-allocation SBC systems at lower bit rates provide different grades of ***communications quality*** encoding. Example 11.5 also illustrates the use of the QMF systems discussed in Section 11.4.

Example 11.3. Sub-band coding of speech at 9.6 kb/s

Table 11.2 shows a sub-band partition for a 9.6 kb/s SBC system. The sub-band sampling rates are obtained from an original sampling rate of 9.6 kHz by using ξ_k:1 decimations, with ξ_k values of 20, 10, 9 and 5.

The PCM coders use adaptive quantization (for example, the AQB system with a one-word memory, Chapter 4); and ranges of step size are individually matched to the long-time-averaged variances of individual sub-band signals. This is indicated by the different Δ_{min} values in Table 11.2. Respective Δ_{max} values are typically 256 times greater.

Table 11.2 Sub-band coder designs for 9.6 kb/s [Crochiere, 1977]

Band	ζ_k Decimation Factor from 9.6 kHz	Band Edges (Hz)	Sub-band Sampling Rates (Hz)	Relative Δ_{min} Values (dB)	R_k (bits/sample)	I_k (kb/s)
1	20	240-480	480	0.0	3	1.44
2	10	480-960	960	−3.0	3	2.88
3	9	1067-1600	1067	−8.5	2	2.13
4	5	1920-2880	1920	−14.0	1.5	2.88
Sync						0.27
			Total Bit Rate I (kb/s)			9.60
			Typical SNR (dB)			10.8

The bit allocation of (3, 3, 2, 1.5) bits/sample is a perceptually optimized design; it codes lower frequency sub-bands with greater fidelity than the higher frequency sub-bands. The fractional value of $R_4 = 1.5$ bits/sample can be obtained in several ways [Crochiere, 1977]; for example, by the use of $R_4 = 1.0$ and $R_4 = 2.0$ for alternate samples, with appropriate modification of step size logics.

Notice that in Table 11.2, the transmission rate for each band is a rational fraction of total rate I so that sub-band data can be multiplexed into a repetitive framed sequence. The lowest common denominator of these rational fractions, including the fraction of transmission rate reserved for frame synchronization purposes (denoted by "Sync" in Table 11.2) determines the smallest possible frame size for the SBC transmission system [Crochiere, 1977]. In certain applications such as voice storage and computer voice response, synchronization procedures may be built-in, and there may not be any need for transmitting special "Sync" bits. This is indeed the situation assumed in Table 11.4.

The quality of the time-invariant 9.6 kb/s system is limited by a reverberant quality in the output $y(n)$ due to inter-band gaps. Designs without gaps involve smaller values of R_k, greater quantization noise, and even lower quality. On the other hand, 9.6 kb/s SBC systems, as a class, perform much better than full-band fixed-prediction speech coders at 9.6 kb/s; in particular, better than 9.6 kb/s ADM systems. In subjective tests, the 9.6 kb/s SBC system is judged to be equivalent to an ADM system at twice the coding rate, i.e., 19.2 kb/s [Crochiere, 1977]. This advantage is directly related to variable bit allocation, and the fact that the quantization noise in the coding of sub-band k with R_k bits/sample is contained within that band. •

Example 11.4. SBC coding of speech at 16 kb/s

Table 11.3 refers to a 16 kb/s SBC system. The explanation of the numbers in this table is very similar to that for Table 11.2. In perceptual tests, this coder is comparable to 24 kb/s DPCM-AQB. The quality of 16 kb/s SBC with fixed bit allocation is useful for many communications applications, although this SBC output is clearly distinguishable from the original input in side-by-side comparisons.

A comparison of the Δ_{min} values in Tables 11.2 and 11.3 shows that the design of Table 11.3 is a better match to the long-time-averaged psd of speech. Note that in Figure 2.9(a), the psd peaks at a value of frequency that is non-zero, and so does the Δ_k versus k profile in Table 11.3. The behavior with respect to frequency k is monotonic in Table 11.2. •

Table 11.3 Sub-band coder design for 16 kb/s [Crochiere, 1977]

Band	ζ_k Decimation Factor from 10.67 kHz	Band Edges (Hz)	Sub-band Sampling Rates (Hz)	Relative Δ_{min} Values (dB)	R_k (bits/sample)	I_k (kb/s)
1	30	178-356	356	−2.0	4	1.42
2	18	296-593	593	0.0	4	2.37
3	10	533-1067	1067	−6.0	3	3.20
4	5	1067-2133	2133	−11.5	2	4.27
5	5	2133-3200	2133	−18.0	2	4.27
Sync						0.47
			Total Bit Rate I (kb/s)			16.00
			Typical SNR (dB)			13.6

Example 11.5. SBC coding of speech at 16, 24 or 32 kb/s

Table 11.4 shows a general sub-band partitioning framework that can be used for SBC coding at bit rates of 16, 24 or 32 kb/s. The sub-band partitioning is obtained by repeated use of the QMF partitioning principle (Problem 11.2). Note that the 16 and 24 kb/s systems use four sub-bands while the 32 kb/s system uses five sub-bands. The speech bandwidth allowed in this five-band system includes the 3 to 4 kHz range; this is a capability in excess of the 3.2 kHz limit assumed for telephone speech.

Figure 11.8 shows sub-band signals $x_k(n)$; $k = 1,2,3,4$ in a 4-band system which uses the four lower bands of Table 11.4. As shown in the table, respective sampling rates are $f_{sk} = 1000$, 1000, 2000 and 2000 Hz. These sampling rates are obtained from an original sampling rate of $f_s = 8000$ Hz by using ζ_k:1 decimations. Respective values of ζ_k, also listed in the table, are 8, 8, 4 and 4. Note that as we move the lowest sub-band (0-500 Hz) to the highest (2000-3000 Hz), the left half of the waveform becomes increasingly more prominent. This is because of the preponderance of high frequency energy in the left half of the original full-band waveform. Notice also that the waveforms in the lower two sub-bands do not look very different. This is because the original full-band waveform in this example has significant energies centered at about 700 Hz, and this is a strong common component in both of the lower two sub-bands, which happen to overlap significantly in the 700 Hz region.

Table 11.4 Sub-band coder designs for 16, 24 or 32 kb/s [Crochiere, 1981].

Band	ζ_k Decimation Factor From 8 kHz	Band Edges (Hz)	Sub-band Sampling Rates (Hz)	R_k (bits/sample) for I (kb/s) = 16	24	32
1	8	0-500	1000	4	5	5
2	8	500-1000	1000	4	5	5
3	4	1000-2000	2000	2	4	4
4	4	2000-3000	2000	2	3	4
5	4	3000-4000	2000	0	0	3

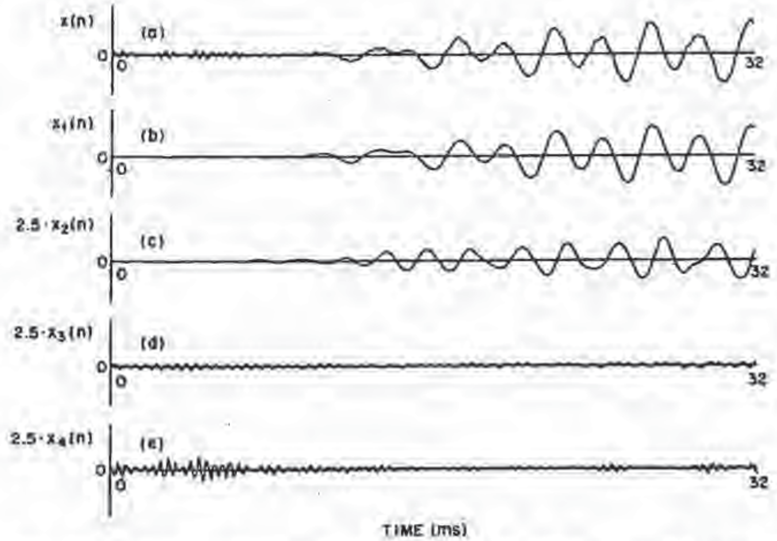


Figure 11.8 (a) Full-band speech waveform of duration 32 ms; and (b)(c)(d)(e) sub-band signals $x_k(n)$ for $k = 1,2,3,4$ in the SBC system of Table 11.4. The sub-band signals $x_2(n)$, $x_3(n)$ and $x_4(n)$ have been magnified by a factor of 2.5 [Cox, 1983].

The full-band signal corresponding to $x_k(k)$ waveforms is a voiced speech segment of duration 32 ms. At the respective Nyquist sampling rates f_{sk}, the sub-band signals exhibit much lower values of adjacent-sample correlation than a full-band speech waveform sampled at its Nyquist rate f_s. The entire waveform $x_1(n)$ in Figure 11.8(b), for example, is sampled only 32 times because of the decimation to 1 kHz. The lower correlations in sub-band signals can also be conjectured from Figure 11.2 where individual sub-bands are seen to have flatter psd's than the full-band speech. (Also, see Problem 11.7). This is the reason why PCM, rather than DPCM, is often assumed in SBC systems, although, with real speech spectra, non-negligible values of prediction gain can be realized even in the case of (individually optimized) fixed predictors of order one.

The nature of the long-time-averaged speech spectrum [Figure 2.9(a)] suggests that the long-term sub-band spectrum can be high-pass in the ranges of sub-bands 1 and 2 of Table 11.4. High-pass spectra in sub-bands can also result from high-frequency pre-emphasis in telephone systems. In fact, in an implementation of the coders of Table 11.4 with telephone speech that includes high-frequency pre-emphasis, the sub-band coders use DPCM-AQB with fixed first-order predictors that reflect highpass spectra in four out of five sub-bands. Recommended values of coefficient h_1, for pre-emphasized speech, are −0.71, −0.28, −0.31, 0.26 and −0.64 in sub-bands 1 through 5, respectively [Daumer, 1982]. •

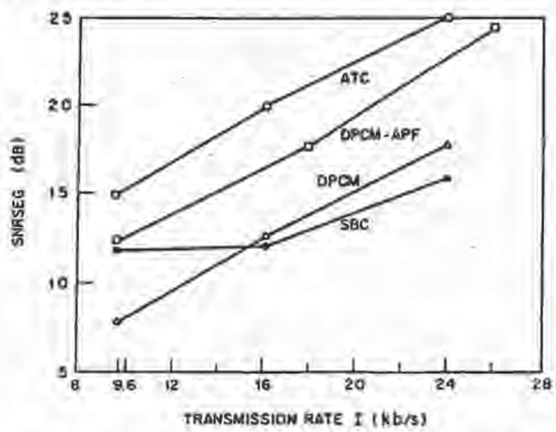
With fixed bit allocation, high quality coding with SBC requires a total bit rate in the order of 32 kb/s. In formal subjective testing, the SBC system of Table 11.4 obtains MOS scores (on a scale of 1 to 5), of 4.3, 3.9 and 3.1 at bit rates of 32, 24 and 16 kb/s [Daumer, 1982]. If the bit rate is at least 24 kb/s, there is an adequate margin of safety for accommodating tandem encodings of 2 to 3 SBC stages with useful final quality.

Although the results of Tables 11.2 to 11.4 refer to specific speech coding systems, they provide general design features that apply to other coding applications as well. An important example is in 9.6 kb/s SBC-TDHS (*time-domain harmonic scaling*) coding systems involving time-compression of input speech by a factor 2:1 prior to sub-band coding at a bit rate slightly under 2(9.6) = 19.2 kb/s [Malah, 1979] [Malah, Crochiere and Cox, 1981] [Crochiere, Cox and Johnston, 1982]. This procedure can result in an improvement over direct SBC at 9.6 kb/s, but the improvement is obtained at the cost of additional complexity, needed for pitch computations that TDHS is based on. Another example of SBC application is the coding of FM-grade audio signals ($W = 20$ kHz) (Problem 11.3) and AM-grade audio signals ($W = 7$ kHz). For example, a two-band SBC system with the band partition [0-3500 Hz] [3500 Hz-7000 Hz] is a good digitizer for 7 kHz material at a coding rate of 56 kb/s [Johnston and Crochiere, 1979].

Comparison of DPCM, DPCM-AP, SBC and ATC. Figure 11.9 shows the performance of four speech coders as a function of bit rate I. The set of coders includes two time-domain coders (DPCM coders with and without adaptive prediction (prediction order equal to one and eight, respectively) and two frequency-domain coders, SBC and ATC (Chapter 12).

One significant comparison in the figure is between the *medium complexity* techniques, SBC (with fixed bit allocation) and DPCM (with fixed prediction). The DPCM coder degrades rapidly if $I < 16$ kb/s ($R < 2$ bits/sample) because of poor quantization performance, compounded by the effects of quantization error feedback. The other significant comparison is between the *high complexity* techniques, DPCM-AP (APC) and ATC. In this specific study, neither of the two techniques incorporated pitch structure. It is interesting that ATC has a consistent

Figure 11.9 Segmental SNR in time-domain and frequency-domain speech coders as a function of bit rate I [Tribolet et al.,©1979, AT&T].



2 to 3 dB advantage. In another study, involving a subjective comparison of APC and ATC, both using pitch in formation, APC was in fact rated slightly better [Daumer, 1982]. Part of the reason for the very good performance of that APC system was a carefully designed algorithm for noise shaping.

The comparisons between DPCM and DPCM-APF, and between SBC and ATC, are less interesting in that the inferences are well expected in each case. The ATC system can be regarded crudely as the equivalent of an SBC system with two very significant refinements: (a) a much larger number of sub-bands M, and (b) adaptive bit-allocation $\{R_k\}$ that is matched to short-term rather than long-term-averaged speech spectrum. As a consequence of these refinements, ATC systems utilize the spectral non-flatness in speech more completely and adaptively, in a fashion similar to redundancy-removal in a full-fledged APC system with higher-order spectrum prediction. The nonadaptive SBC system utilizes spectral non-flatness only in a non-adaptive, and hence incomplete sense. A special, but academic, input for which SBC provides complete redundancy removal is one with a staircase-type psd, as in Figure 2.24. For that two-level psd example, SBC with $M = 2$ provides a complete utilization of spectral non-flatness (Example 11.3 and Problem 11.6).

Combinations of sub-band coding and adaptive prediction. The performance of sub-band coding can be significantly improved by supplementing it with the time-domain operation of adaptive prediction [Atal and Remde, 1981] [Honda, Kitawaki and Itakura, 1982]. The general philosophy of these hybrid techniques is to split the burden of redundancy removal between the frequency domain and the time domain. The hybrid approach also provides the possibility of noise shaping in both the frequency domain and the time domain. In one system, the latter is provided by dividing basic time blocks of duration in the order of one pitch period into sub-blocks, typically four in number; and allocating quantization bits on the basis of prediction error energy in each sub-interval. This is done separately for each of typically three to four sub-bands. The bit allocation in the time domain provides better encoding of the higher-valued prediction error samples during the onset of a pitch period and mitigates the problem of quantization error feedback, separately in each sub-band. With spectrum prediction of order four and pitch prediction, the sub-band coder with time-frequency bit allocation provides MOS quality scores in the order of 3.5 at 16 kb/s and 4.0 at 24 kb/s [Honda, Kitawaki and Itakura, 1982].

11.6 Transmission Error Effects

Bit errors in an individual sub-band will generate error contributions at the receiver output within that frequency band. Since the channel error contributions in different sub-bands are expected to be uncorrelated, the corresponding error variances will simply add. The results of Section 11.2 and Section 4.9 can therefore be used to derive an expression for channel error variance σ_c^2 in an SBC system. Also, explicit forms of error-protection can be used to mitigate channel error effects in SBC decoding. Error-protection systems can exploit not only the different sensitivities of various bits in transmitted codewords (as in Section 4.9), but also the different sensitivities of various sub-bands to transmission error effects. These points will be made more quantitative during the discussion of Transform Coding in Chapter 12.

Example 11.6. Transmission error effects in 24 kb/s SBC systems

Figure 11.10 describes the performance of three SBC systems at $I = 24$ kb/s and bit error rate p_e. In coder A, the adaptive quantizers use the one-word memory logic (4.185). In coder B, the adaptive quantizers use the robust version (4.199a) of the adaptation logic, with step size leakage factor $\beta = 31/32$. In coder C, the sign bit (most significant bit) as well as the most significant magnitude bit in the lowest frequency sub-band are ideally error protected. This protection requires an appropriate amount of bit-stealing from the quantizers, in order to maintain the total bit rate at the original value of $I = 24$ kb/s (Problem 11.8). However, the resulting increase of quantization noise is more than compensated for by reductions in channel noise if the error rate p_e is in the order of 10^{-2} or greater. The overall performance is so as to provide acceptable speech outputs at $p_e = 10^{-2}$, but not at $p_e = 10^{-1}$. ●

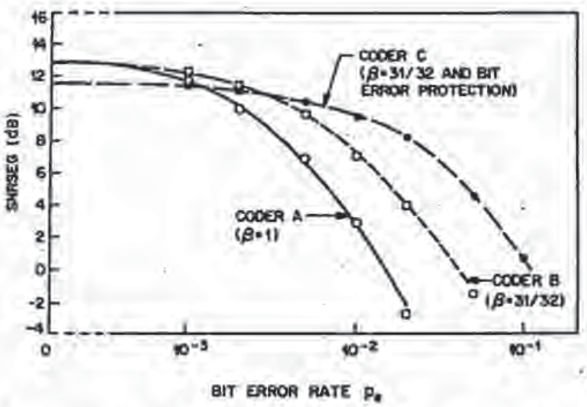


Figure 11.10 Transmission error performance of 16 kb/s SBC systems. Values of segmental SNR as a function of bit error rate p_e [Crochiere, ©1978, AT&T].

Problems

(11.1) Consider a 4-band SBC system where sub-band partitions are constrained by filter-bank considerations to be either (a) or (b) below:

(a) [0–800] [800–1600] [1600–2400] [2400–3200] Hz

(b) [225–450] [450–900] [1000–1500] [1800–2700] Hz

(11.2)

(11.3)

(11.4)

(11.5)

(11.6)

not only the
Section 4.9),
error effects.
of Transform

kb/s and bit
logic (4.185).
aptation logic,
ificant bit) as
e ideally error
ing from the
f $I = 24$ kb/s
is more than
der of 10^{-2} or
at $p_e = 10^{-3}$.

ental SNR as a

onstrained by

Ignoring side information for synchronization bits, show that bit allocations that provide 9.6 kb/s with the filter-banks above are respectively

(a) 3, 2, 1, 0 bits/sample

(b) 4, 3, 2, 1 bits/sample

Note that (b) is closer to the design recommended in Table 11.2.

(11.2) It is required to design a four-band filter-bank [0-0.5] [0.5-1.0] [1.0-2.0] [2.0-3.0] kHz for 16 kb/s coding of speech.

(a) Show the band-division tree that realizes this partition using repeated quadrature mirror filtering of the [0-4.0] kHz band.

(b) Assuming that $\max\{R_k\} = 5$ and $\min\{R_k\} = 2$ bits/sample, show that there are at least 2 bit assignments [5,3,2,2] [4,4,2,2] that realize a total bit rate $I = 16$ kb/s. Ignore side information considerations, as in Problem 11.1.

(11.3) Consider a 6-band SBC system for studio-grade music, with sub-bands [0-0.625] [0.625-1.25] [1.25-2.5] [2.5-5.0] [5.0-10.0] [10.0-20.0] kHz. Allowing 6 kb/s for synchronization information, show that the bit allocation (12, 10, 9, 8, 6, 5) bits/sample permits digital transmission over a 256 kb/s channel.

(11.4) Consider a SBC system with M bands of equal width ΔW. Consider the SBC input as a 1-second sequence of $2M\ \Delta W$ samples. Denote variances of individual samples of sub-band k by σ_{xk}^2 ; $k = 1,2,...,M$. Let each of these samples in sub-band k be quantized using R_k bits/sample. Because of a constant sampling rate $2\Delta W$, the total number of samples per second is independent of k. The optimum bit allocation for minimum mse, and the maximum gain G_{SBC} over single-band PCM, are given (on a *per sample* basis) by (11.10) and (11.12).

It is also useful to give results on a *per second* basis. Define $\mathbf{R}$ as the total bit rate (bits/sec) in the SBC system, $\mathbf{R}_k = 2\Delta W R_k$ as the bit rate (bits/second) allocated to encode band k, and $\sigma_{zk}^2 = \sigma_{xk}^2 2\Delta W$ as the power in sub-band k. Show that the formulas (11.10) and (11.12) are equivalent to the pair of formulas

$$\mathbf{R}_{k,opt} = \frac{\mathbf{R}}{M} + \Delta W \log_2 \frac{\sigma_{zk}^2}{\left[\prod_{k=1}^{M}\sigma_{zk}^2\right]^{1/M}} ; \qquad \max\{G_{SBC}\} = \frac{\frac{1}{M}\sum_{k=1}^{M}\sigma_{zk}^2}{\left[\prod_{k=1}^{M}\sigma_{zk}^2\right]^{1/M}}$$

(11.5) Refer to Problem 11.4 and the equal-width four-band partition of Problem 11.1(a). Equate the power $\sigma_{zk}^2 = \sigma_{xk}^2\ 2\Delta W$ to $S_k 2\Delta W$ where S_k represents the average value of input psd in sub-band k. Use the long-time psd of speech in Figure 2.9(a) together with the formula for $\mathbf{R}_{k,opt}$ in Problem 11.4 to show that the bit allocations in Problem 11.1(a) are indeed nearly optimal for a mmse criterion.

(11.6) Consider the highly structured two-level psd of Figure 2.24 and (2.168).

(a) Consider SBC with $M = 2\ \Delta\Omega = \Omega_W/2$, and band-edges at 0, $\Omega_W/2$ and Ω_W to follow the shape of the input psd. Use the result of Problem 11.4 to show that the maximum gain in 2-band SBC is

$$\max\{{}^2G_{SBC}\} = [\alpha(2-\alpha)]^{-1/2}$$

(b) Show that $\max\{{}^2G_{SBC}\} = \gamma_x^{-2} = \max\{G_{SBC}\}$, the reciprocal of the spectral flatness measure (Problem 2.22) of the process, i.e., SBC with $M = 2$ bands is sufficient to realize the maximum performance for this special psd.

(11.7) Consider the case of M equal-width sub-bands.

(a) Show that the adjacent sample correlation ρ_{1k} for the decimated signal in sub-band k vanishes if the psd in that sub-band is flat-topped.

(b) Determine the four values of ρ_{1k} for the 4-band partition of a signal with an integrated power spectrum and $\psi = 4$.

(11.8) Consider a 4-band SBC system for 24 kb/s speech coding, with the QMF partition [0, 1000] [1000, 2000] [2000, 3000] [3000, 4000] Hz, and with a normal bit allocation of (5, 4, 2, 1) bits/sample for quantization. Consider that this system is adapted to a noisy channel by protecting all bits in the first sub-band using a (12, 8) Hamming code (with 4 redundant bits for every 8 message bits). Show that if the bit allocation for quantization is changed to (4, 3, 2, 1) bits/sample, the total transmission rate is still 24 kb/s.

References

B. S. Atal and J. R. Remde, "Split-Band APC System for Low Bit Rate Encoding of Speech," Proc. ICASSP, April 1981.

H. G. Bellanger, G. Bonnerot, and M. Coudreuse, "Digital Filtering by Polyphase Network: Application to Sample-Rate Alteration and Filter-Banks", IEEE Trans. on Acoustics, Speech and Signal Proc., pp. 252-259, 1976.

R. V. Cox, unpublished work, Bell Laboratories, 1981.

R. V. Cox, "A Comparison of Three Speech Coders to Be Implemented on the Digital Signal Processor," Bell System Tech. J., pp. 1411-1421, September 1981.

R. V. Cox, unpublished work, Bell Laboratories, 1983.

R. E. Crochiere, "On the Design of Sub-Band Coders for Low Bit Rate Speech Communication," Bell System Tech. J., pp. 747-771, May-June 1977.

R. E. Crochiere, "An Analysis of 16 kb/s Sub-band Coder Performance: Dynamic Range, Tandem Connections, and Channel Errors," Bell System Tech., J., pp. 2927-2952, October 1978.

R. E. Crochiere, "Sub-band Coding," Bell System Tech. J., pp. 1633-1654, September 1981.

R. E. Crochiere, R. V. Cox and J. D. Johnston, "Real Time Speech Coding," IEEE Trans. on Communications, pp. 621-634, April 1982.

R. E. Crochiere and L. R. Rabiner, "Interpolation and Decimation of Digital Signals — A Tutorial Review," Proc. IEEE, pp. 300-331, March 1981.

R. E. Crochiere and L. R. Rabiner, *Multirate Digital Processing*, Prentice Hall, 1983.

R. E. Crochiere and M. R. Sambur, "A Variable-Band Coding Scheme for Speech Encoding at 4.8 kb/s", Bell System Tech. J., pp. 771-780, May-June 1977.

R. E. Crochiere, S. A. Webber and J. L. Flanagan, "Digital Coding of Speech in Sub-bands," Bell System Tech. J., pp. 1069-1085, October 1976.

W. R. Daumer, "Subjective Evaluation of Several Efficient Speech Coders," IEEE Trans. on Communications, pp. 662-665, April 1982.

D. Esteban and C. Galand, "Application of Quadrature Mirror Filters to Split Band Voice Coding Schemes," Proc. ICASSP, pp. 191-195, May 1977.

D. Esteban and C. Galand, "32 kb/s CCITT-Compatible Split Band Coding Scheme," Proc. ICASSP, pp. 320-325, 1978.

J. L. Flanagan et al., "Speech Coding", IEEE Trans. on Communications, pp. 710-737, April 1979.

C. Galand and D. Esteban, "16 kb/s Sub-band Coder Incorporating Variable Overhead Information," Proc. ICASSP, pp. 1684-1687, April 1982.

L. M. Goodman, "Channel Encoders," Proc. IEEE, pp. 127-128, 1967.

C. Grauel, "Sub-band Coding with Adaptive Bit Allocation," Signal Proc., 1980.

C. D. Heron, R. E. Crochiere and R. V. Cox, "A 32-Band Sub-Band/Transform Coder Incorporating Vector Quantization for Dynamic Bit Allocation," Proc. ICASSP, pp. 1276-1279, April 1983.

M. Honda, N. Kitawaki and F. Itakura, "Adaptive Bit Allocation Scheme in Predictive Coding of Speech," Proc. ICASSP, pp. 1672-1675, May 1982.

J. D. Johnston, "A Filter Family Designed for Use in Quadrature Mirror Filter Banks," Proc. ICASSP, pp. 291-294, April 1980.

J. D. Johnston and R. E. Crochiere, "An All-Digital 'Commentary-Grade' Sub-band Coder," Audio Eng. Soc. Conv., AES preprint, May 1979.

D. Malah, "Time Domain Harmonic Scaling Algorithms for Bandwidth Reduction and Time Scaling of Speech Signals," IEEE Trans. on Acoustics, Speech and Signal Processing, pp. 121-133, April 1979.

D. Malah, R. E. Crochiere and R. V. Cox, "Performance of Transform and Sub-Band Coding Systems Combined with Harmonic Scaling of Speech," IEEE Trans. on Acoustics, Speech and Signal Proc., pp. 273-283, April 1981.

L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, N.J., 1978.

T. A. Ramstad, "Sub-band Coder with a Simple Bit Allocation Algorithm: A Possible Candidate for Digital Mobile Telephony," Proc. ICASSP, pp. 203-207, May 1982.

T. A. Ramstad and O. Foss, "Sub-band Coder Design Using Recursive Quadrature Mirror Filters," Proc. EUSIPCO, pp. 747-752, 1980.

J. E. Stjernvall, "On Rate and Frequency Allocation in Sub-band Coding of Gaussian Sources", Dept. of Elec. Eng. Report No. 1977-12-05, Linkoping University, Sweden, 1977.

J. M. Tribolet, P. Noll, B. J. McDermott and R. E. Crochiere, "A Comparison of the Performance of Our Low Bit Rate Speech Waveform Coders," Bell System Tech. J., pp. 699-713, March 1979.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

Techniques for data hiding

PD: '95
pp. 164 - 173 = 10

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.

0-8194-1767-X/95/$6.00

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.

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

The *expected* 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 95% 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.

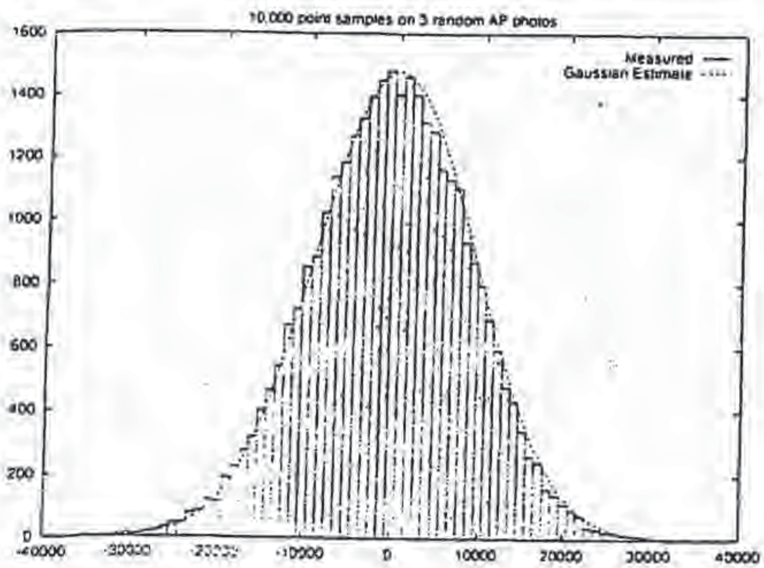


Figure 1: Experimental results from Patchwork. The sum S is plotted along the horizontal axis. The number of trials that yield S is plotted along the vertical axis. The *expected* value of S is 24.79 where n=10,000.

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™ (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:

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

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

$$(\dot{\phi}_1(\omega_k) = \dot{\phi}_0(\omega_k) - \Delta\dot{\phi}_1(\omega_k))$$

$$\ldots$$

$$(\dot{\phi}_i(\omega_k) = \dot{\phi}_{i-1}(\omega_k) + \Delta\dot{\phi}_i(\omega_k))$$

(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.

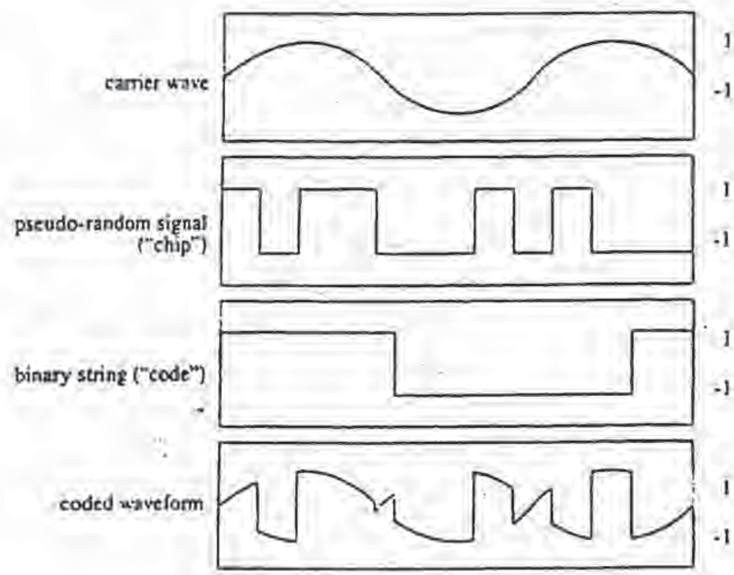


Figure 2: Synthesized spread spectrum information encoded by the Direct Sequence method.

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:

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

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.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☑ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

EMBEDDING ROBUST LABELS INTO IMAGES FOR COPYRIGHT PROTECTION

Jian Zhao & Eckhard Koch
Fraunhofer Institute for Computer Graphics
Wilhelminenstr. 7, 64283 Darmstadt, Germany
Email: {zhao, ekoch}@igd.fhg.de

Abstract

This paper describes a set of novel steganographic methods to secretly embed robust labels into image data for identifying image copyright holder and original distributor in digital networked environment. The embedded label is undetectable, unremovable and unalterable. Furthermore it can survive processing which does not seriously reduce the quality of the image, such as lossy image compression, low pass filtering and image format conversions.

1 Introduction

The wide use of digitally formatted audio, video and printed information in network environment has been slowed down by the lack of adequate protection on them. Developers and publishers hesitate to distribute their sensitive or valuable materials because of the easiness of illicit copying and dissemination [3],[6],[7].

Compared to ordinary paper form information, digitized multimedia information (image, text, audio, video) provides many advantages, such as easy and inexpensive duplication and re-use, less expensive and more flexible transmission either electronically (e.g. through the Internet) or physically (e.g. as CD-ROM). Furthermore, transferring such information electronically through network is faster and needs less efforts than physical paper copying, distribution and update. However, these advantages also significantly increase the problems associated with enforcing *copyright* on the electronic information.

Basically, in order to protect distributed electronic multimedia information, we need two types of protections. First, the multimedia data must contain a label or code, which identifies it uniquely as property of the copyright holder. Second, the multimedia data should be marked in a manner which allows its distribution to be tracked. This does not limit the number of copies allowed (vs. copy protection), but provides a mean to check the original distributor. In order to prevent any copyright forgery, misuse and violation, the copyright label must be unremovable and unalterable, and furthermore survive processing which does not seriously reduce the quality of the data. This requires that first the label must be secretly stored in a multimedia data, i.e. the locations for embedding this label are secret, second the label must be robust even if the labeled multimedia data has been processed incidentally or intentionally.

This paper describes a set of novel steganographic methods to secretly embed robust labels into image data for copyright protection in open networked environment. The label embedded in the image can be assigned or generated in a way that it is able to identify the copyright holder and the original purchaser (distributor).

Steganographic method is a technique embedding additional information into a data by modifying the original data without affecting the quality of the data. Many steganographic methods have been proposed to aim at storing additional information to identify or label formatted electronic documents [1], images, video [8], and audio data. However, they are far away from the requirements in protecting multimedia information in a networked environment, because although some of them provide secret locations for label embedding, none of them is able to prevent attacks on the embedded information by simple image processing, i.e. they do not adequately address the possibilities of using data compression, low pass filtering and/or simply changing the file format to remove an embedded code.

The discussion begins with a general framework for copyright label embedding. Then two specific methods are developed: one is based on the JPEG compression model for embedding labels in gray-scaled and color images, and the other is based on the black/white rate for binary images. Finally, these methods are tested experimentally and the future work is discussed.

2 Robust Label Embedding Framework

The system developed along the methods presented in this paper is called 'SysCoP' (System for Copyright Protection). It consists of a set of methods to embed robust labels into different types of images. Currently, the system supports gray-scaled, color, and binary images. These methods share an algorithm framework for both label writing and reading described below.

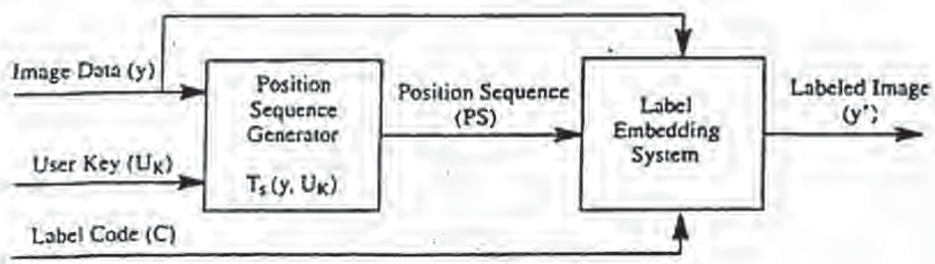


Figure 1. Write label

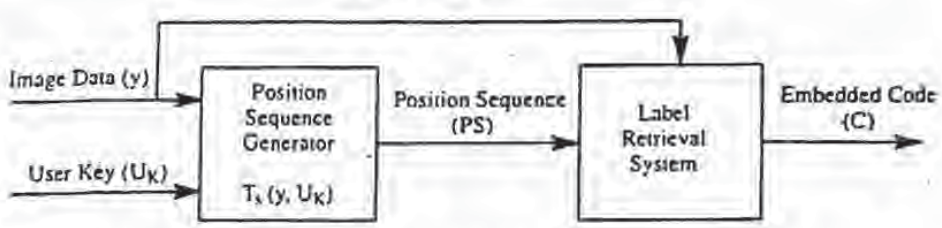


Figure 2. Read label

The framework, as shown in Figure 1 for label writing and Figure 2 for label reading, is composed of two steps. The first step generates a pseudo random position sequence for selecting blocks where the code is embedded. This step is denoted as a function $T_s(y, U_K)$ where y is the image data to be labeled, and U_K is the user-supplied secret key. The second step simply embeds or retrieves the code into or from the blocks specified in the position sequence. The methods for embedding or reading code depend on types of images, and will be described individually in the next section.

The function $T_s(y, U_K)$ firstly extracts some features from the image data and then use them together with the user secret key as the seeds for position sequence generation [4]. Ideally, the features of the image data used here must meet the following requirements:

- they must be robust against simple image processing that does not affect the visual quality of the image, and
- they must be image-dependent, i.e. the image can be identified, uniquely in an ideal case, by these features extracted from the image data.

However, to achieve the contradictory requirements above, in fact, is a very hard work. Much research has been done in related fields for other purposes, e.g. content-based image retrieval, image segment and pattern recognition, which can be found in many literature (e.g. [9]). Currently, we only use the width and height of an image for the position generation in the function $T_s(y, U_K)$.

Before describing the framework, we introduce some terminology. A *block* of an image consists of 8x8 pixels. In this framework, it is either a contiguous or a distributed block. A *contiguous block* is a 8x8 square in an image component. A *distributed block* is a collection of 8x8 pixels each of which is selected randomly from the whole image space. The purpose of distributed block is to prevent or discourage attackers from detecting embedding locations by comparing different labeled images. A block is 'invalid' for code embedding if too big modifications to the block data are needed in order to embed a bit into this block. The criteria of validation of the block depends on the specific label-embedding methods to be described in the next section.

Let *C* be the embedded code, and represented as a binary bit stream $\{c_0, c_1, ..., c_n\}$. Let *i* be the index of current bit in this stream. Let $\mathfrak{B}$ be the block set in which each block has been randomly selected. Initialize i to 0 and $\mathfrak{B}$ to $\{\}$. The framework for writing and reading robust labels is described below in Algorithm 1(a)-(b).

Algorithm 1(a): Framework (write).

(1) If $i \geq n$, return.

(2) Randomly select a block *b*, using the position sequence generation function $T_s(U_K, y)$ in Figure 1.

(3) If *b* exists already in $\mathfrak{B}$, go to (2), otherwise add *b* to $\mathfrak{B}$.

(4) Call *check_write(b, c_i)* to check wether *b* is a valid block; if this function returns False (i.e. the block *b* is an invalid block), go to (2).

(5) Call *write(b,c_i)* to embed a bit c_i to the block *b*.

(6) Increment i, go to (1).

Algorithm 1(b): Framework (read).

(1) If $i \geq n$, return.

(2) Randomly select a distributed or a contiguous 8x8 block *b*, using the position sequence generation function $T_s(U_K, y)$ in Figure 2.

(3) If *b* exists already in $\mathcal{B}$, then go to (2), otherwise add *b* to $\mathcal{B}$.

(4) Call ***check_read(b,c_i)*** to check wether *b* is a valid block: if this function returns False (i.e. the block *b* is an invalid block), go to (2).

(5) Call ***read(b)*** to retrieve a bit from the block *b*.

(6) Increment i, and go to (1).

3 Robust Label Embedding Methods

3.1 JPEG-Based Label Embedding for Gray-Scaled and Color Images

In this subsection, we first introduce briefly the JPEG compression model, then describe the principle of the embedding methods based on the JPEG compression model. Finally, the algorithms for embedding labels into gray-scaled and color images are developed.

Suppose the source image composes three components: one luminance (Y) and two chrominance (I and Q). That is, each pixel in the image can be represented with a triple of 8-bit values (Y,I,Q). Each component is broken up into contiguous blocks. The JPEG compression consists of six steps: normalization, DCT transformation, quantization, zigzag scan, run-length encoding and Huffman coding steps. Since our method is applied after the quantization step, we only describe briefly the first three steps of the JPEG model. The detailed description of the JPEG model is available elsewhere [11].

The normalization step brings all image values into a range, e.g. between −128 and 127 for a 24-bit image. The DCT step applies the discrete cosine transform (DCT) to each 8x8 block, producing a new 8x8 block [10]. If we call the new block Y(k,l), with $k,l \in 0..7$, the equation of the DCT is:

$$Y(k, l) = \frac{1}{4} \sum_i \sum_j C(i, k)C(j, l)y[i, j]$$

where

$$C(i, k) = A(k)\frac{\cos(2i + 1)k\pi}{16} \qquad A(k) = \frac{1}{\sqrt{2}} \text{ for } k = 0, \ A(k) = 1 \text{ for } k \neq 0 \qquad (1)$$

Each element of the new block is further quantized:

$$Y_Q[k, l] = \mathrm{Round}\left(\frac{Y[k, l]}{q[k, l]}\right) \qquad (2)$$

Equation (2) represents the entire lossy modelling process of the JPEG compression. The choice of the quantization table (q[k,l]) determines both the amount of compression and the quality of the decompressed image. The JPEG standard includes recommended luminance and chrominance quantization tables resulting from human factors studies. To obtain different compression quality, we typically use a *quality factor* to scale the values of these default quantization tables.

In the JPEG decompression process, each element of $Y_Q(k,l)$ is multiplied by $q(k,l)$ to recover an approximation of $Y(k,l)$. Finally, the image block $y(i,j)$ can be recovered by performing an inverse 2-D DCT (IDCT):

$$y(i,j) = \frac{1}{4}\sum_k \sum_l C(i,k)C(j,l)Y[k,l] \qquad (3)$$

The basic principle of the JPEG-based embedding method is that quantized elements have a moderate variance level in the middle frequency coefficient ranges, where scattered changes in the image data should not be noticeably visible. The specific frequencies being used to embed the code will be 'hopped' in this range to increase the robustness of the signal and making it more difficult to find [5],[2]. A label bit is embedded through holding specific relationship among three quantized elements of a block. The relationships among them compose 8 patterns (combinations) which are divided into three groups: two of them are used to represent '1' or '0' for embedded codes (valid patterns), and the other represents *invalid patterns*. If too big modifications are needed to hold a desired valid pattern representing a bit, this block is invalid. In this case, the relationships among the three elements of the selected location set are modified to any of the invalid patterns to 'tell' the label-retrieval process that this block is invalid. The criterion for invalid blocks is specified by a parameter MD, i.e. the maximum difference between any two elements of a selected location set in order to reach the desired valid pattern.

Set No.	(k_1,l_1)	(k_2,l_2)	(k_3,l_3)
1	2(0,2)	9(1,1)	10(1,2)
2	9(1,1)	2(0,2)	10(1,2)
3	3(0,3)	10(1,2)	11(1,3)
4	10(1,2)	3(0,3)	11(1,3)
5	9(1,1)	2(0,2)	10(1,2)
6	2(0,2)	9(1,1)	10(1,2)
7	9(1,1)	16(2,0)	2(0,2)
8	16(2,0)	9(1,1)	2(0,2)
9	2(0,2)	9(1,1)	16(2,0)
10	9(1,1)	2(0,2)	16(2,0)
11	10(1,2)	17(2,1)	3(0,3)
12	17(2,1)	10(1,2)	3(0,3)
13	10(1,2)	3(0,3)	17(2,1)
14	3(0,3)	10(1,2)	17(2,1)
15	9(1,1)	16(2,0)	17(2,1)
16	16(2,0)	9(1,1)	17(2,1)
17	10(1,2)	17(2,1)	18([illegible]
18	17(2,1)	10(1,2)	18(2,2)

Table 1. Possible location sets

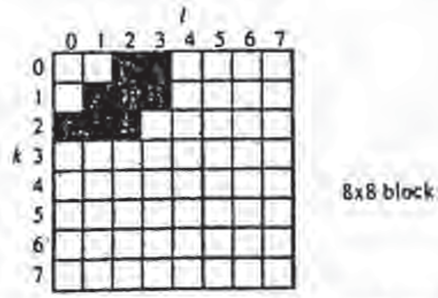


Figure 3. Possible locations for embedding code in a block

(k_1,l_1)	(k_2,l_2)	(k_3,l_3)	
H	M	L	patterns for '1'
M	H	L	
H	H	L	
M	L	H	patterns for '0'
L	M	H	
L	L	H	
H	L	M	invalid patterns
L	H	M	
M	M	M	

Table 2. '1', '0' and invalid patterns (H: High, M: Middle, L: Low)

Our statistic results of the possible locations holding the specific frequencies are illustrated in Figure 3 as shadowed areas within a 8x8 block. In Table 1, we give our statistic results of the best location sets combined from these possible elements. The algorithms to write and read a label into and from an color or gray-scaled image are described in Algorithm 2(a)-(d).

Two parameters are provided for adjusting the robustness vs. modification visibility in an embedding process. The first one is the distance (D) between selected quantized frequency coefficients for representing an embedded bit. The default value of this distance is 1. The greater distance produces stronger robustness, but also may cause more serious modification visibilities. The second parameter is the quantization factor (Q) used to quantize the values selected for embedding code. The greater quantization factor results in less modifications to image data but weaker robustness against lossy JPEG compression. The default value of this quantization factor is 75%.

Algorithm 2(a): check_write(b,c_i)

(1) A three-element location set of the block *b* is pseudo-randomly (from the user key and image data) selected from the possible location sets listed in Table 1. They are denoted as (k_1,l_1), (k_2,l_2) and (k_3,l_3).

(2) The block *b* is locally DCT transformed and quantized at the locations (k_1,l_1), (k_2,l_2) and (k_3,l_3) with the quality factor Q parameter. Let $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$ and $Y_Q(k_3,l_3)$ be the quantized coefficient values at the selected locations.

(3) When c_i=1, if MIN ($|Y_Q(k_1,l_1)|$, $|Y_Q(k_2,l_2)|$) + MD < $|Y_Q(k_3,l_3)|$ where $|Y_Q(k_u,l_u)|$ is the absolute value of $Y_Q(k_u,l_u)$ with u ∈ 1..3, MIN is an operation that returns the minimum value of two elements, and MD is the maximum modification distance, then *b* is an invalid block:

(i) modify them to any of the invalid patterns shown in Table 2,

(ii) de-quantize and inversely transform (IDCT) them, and write them back to the block *b*,

(iii) return False.

(4) When c_i=0, if MAX ($|Y_Q(k_1,l_1)|$, $|Y_Q(k_2,l_2)|$) > $|Y_Q(k_3,l_3)|$ + MD where MAX is an operation that returns the maximum value of two elements, and MD is the maximum modification distance, then *b* is an invalid block:

(i) modify them to any of the invalid patterns shown in Table 2,

(ii) de-quantize, inversely transform (IDCT) them, and write them back to the block *b*,

(iii) return False.

(5) For other cases, return True.

Algorithm 2(b): check_read(b, c_i)

(1) A three-element location set of the block *b* is pseudo-randomly (from the user key and image data) selected from the possible location sets listed in Table 1. They are denoted as (k_1,l_1), (k_2,l_2) and (k_3,l_3).

(2) The block *b* is locally DCT transformed and quantized at the locations (k_1,l_1), (k_2,l_2) and (k_3,l_3) with the quality factor Q parameter. Let $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$ and $Y_Q(k_3,l_3)$ be the quantized coefficient values at the selected locations.

(3) If $|Y_Q(k_1,l_1)|$, $|Y_Q(k_2,l_2|$ and $|Y_Q(k_3,l_3)|$ form any of the invalid patterns as illustrated in Table 2, return False, otherwise return True.

Algorithm 2(c): write(b, c_i)

Assume that a valid three-element location set of the block *b* has been pseudo-randomly selected. They are denoted as (k_1,l_1), (k_2,l_2), and (k_3,l_3). The block *b* is locally DCT transformed, and quantized at the locations (k_1,l_1), (k_2,l_2) and (k_3,l_3) with the quality factor Q. Let $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$, and $Y_Q(k_3,l_3)$ be the quantized coefficient values at the selected locations.

(1) When $c_i=1$, modify the $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$ and $Y_Q(k_3,l_3)$ such that they satisfy the following conditions: $Y_Q(k_1,l_1) > Y_Q(k_3,l_3) + D$, and $Y_Q(k_2,l_2) > Y_Q(k_3,l_3) + D$

(2) When $c_i=0$, modify the $Q(k_1,l_1)$, $Q(k_2,l_2)$ and $Y_Q(k_3,l_3)$ such that they satisfy the following conditions: $Y_Q(k_1,l_1) + D < Y_Q(k_3,l_3)$, and $Y_Q(k_2,l_2) + D < Y_Q(k_3,l_3)$

(3) $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$ and $Y_Q(k_3,l_3)$ are de-quantized, inversely transformed (IDCT), and written back to the block *b*.

Algorithm 2(d): read(b)

Assume that a valid three-element location set of the block *b* has been pseudo-randomly selected. They are denoted as (k_1,l_1), (k_2,l_2), and (k_3,l_3). The block *b* is locally DCT transformed, and quantized at the locations (k_1,l_1), (k_2,l_2) and (k_3,l_3) with the quality factor Q. Let $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$, and $Y_Q(k_3,l_3)$ be the quantized coefficient values at the selected locations.

(1) If $Y_Q(k_1,l_1) > Y_Q(k_2,l_2) + D$ and $Y_Q(k_2,l_2) > Y_Q(k_3,l_3) + D$, return 1.

(2) If $Y_Q(k_1,l_1) + D < Y_Q(k_3,l_3)$, and $Y_Q(k_2,l_2) + D < Y_Q(k_3,l_3)$, returns 0.

(3) In other cases, the embedded bit in this block *b* has been damaged.

3.2 Black/White Rate-Based Label Embedding for Binary Images

The value of each pixel in a binary image is either '1' or '0'. This determines that, in general, there is no 'noise' space which can be used for embedding additional information. To do it, we must find appropriate image areas where modifications for embedding labels do not affect seriously the quality of the original image. Obviously, these areas are varied with individual images or at least with types of images.

The proposed method for binary images is based on the ratio of '1' and '0' in a selected block. Suppose '1' represent black bit and '0' represent white bit in the source binary image. Let $P_1(b)$ be the rate (percentage) of blacks in the selected block *b*:

$$P_1(b) = \frac{N_1(b)}{64} \quad \text{where } N_1(b) \text{ is the number of '1' in the block } b.$$

Since the sum of percentages of blacks and whites in a block is 100%, the rate (percentage) of whites in the block *b* is $P_0(b) = 100 - P_1(b)$. A bit is embedded into a block *b* in the following way: a '1' is embedded into the block *b* if $P_1(b)$ is greater than a given threshold, and a '0' is embedded into the block *b* if $P_1(b)$ is less than another given threshold. A sequence of contiguous or distributed blocks is modified by switching whites to blacks or vice versa until such thresholds are reached.

We have classified two categories of binary images on which the generic method described above can be applied. These binary images are identified by distribution feature of blacks and whites. The first type of binary images is dithered image in which the black and white are well interlaced. The second type of binary images is black/white sharply contrasted images in which there exist clear boundaries between black and white areas.

Two modification strategies are adopted for these two types of binary images, respectively. For dithered binary images, modifications are well-distributed throughout the whole block: the bit that has most neighbors with the same value (either black or white) is reversed. For sharply contrasted binary images, modifications are carried out at the boundary of black and white pixels: the bit that has most neighbors with the opposite value is reversed. At the borders of the contiguous block, the neighbor bits in the neighbor blocks are also taken into account in both approaches. Two examples of both modification strategies are illustrated in Figure 4 and 5, respectively.

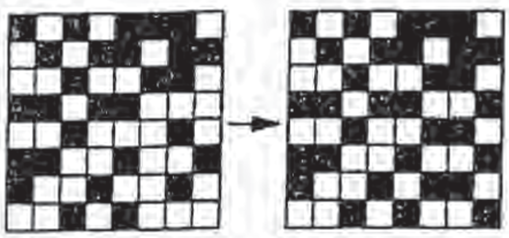


Figure 4. Well-distributed modifications

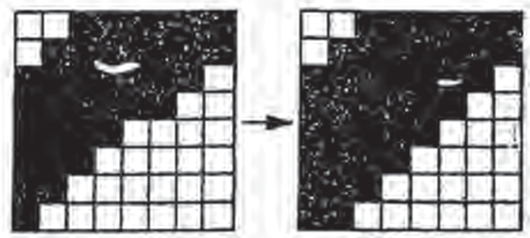


Figure 5. Modifications at black and white boundary

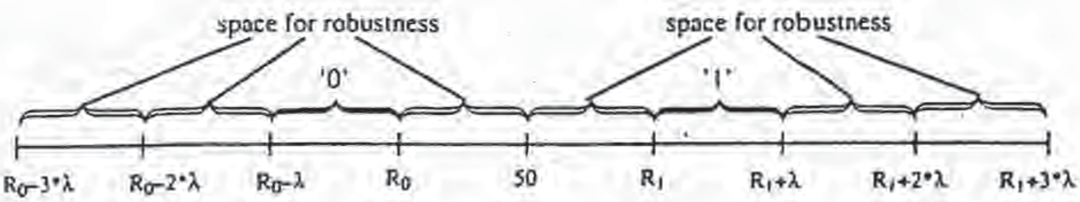


Figure 6. Achieve robustness in the black/white rate-based embedding method

Let R_1 be the threshold rate for '1'. Thus, the threshold rate for '0' is $R_0 = (100\%-R_1)$. Let λ be the robustness degree against image processing of labeled images. It represents the number of bits that can be altered after image processing without damage of embedded bits. For example, when λ is 5%, alternation (i.e. reversion from '1' to '0' or vice versa) of less than 4 bits in a block does not damage the embedded code. Our experiments have shown that the following values of them are the reasonable choices both in robustness of embedding code and the modification visibility:

$R_1 = 55$, $R_0 = 45$, and $\lambda = 5$.

The algorithms to write and read a label into and from a binary image are described in Algorithm 3(a)-(d).

Algorithm 3(a): check_write(b, c_i)

(1) If $P_1(b) > R_1 + 3*\lambda$ or $P_1(b) < R_0 - 3*\lambda$, return False.

(2) When c_i=1, if $P_1(b) < R_0$, modify the block *b* such that

$$P_1(b) < R_0 - 3 \cdot \lambda,$$

and then return False.

(3) When c_i=0, if $P_1(b) > R_1$, modify the block *b* such that

$$P_1(b) > R_1 + 3 \cdot \lambda,$$

and then return False.

(4) For other cases, return True.

Algorithm 3(b): check_read(b, c_i)

(1) If $P_1(b) > R_1 + 2 \cdot \lambda$ or $P_1(b) < R_0 - 2 \cdot \lambda$, return False.

(2) For other cases, return True.

Algorithm 3(c): write(b, c_i)

Assume that a valid block *b* has been pseudo-randomly selected. A bit c_i is embedded into *b* b switching blacks to whites or vise versa using the different modification strategies described above i order to reach a specified threshold rate.

(1) When c_i=1, modify the block *b* such that:

$$P_1(b) \geq R_1 \text{ and } P_1(b) \leq R_1 + \lambda.$$

(2) When c_i=0, modify the block *b* such that:

$$P_1(b) \leq R_0 \text{ and } P_1(b) \geq R_0 - \lambda.$$

(3) write the block *b* back to the image.

Algorithm 3(d): read(b)

Assume that a valid block *b* has been pseudo-randomly selected.

(1) If $P_1(b) > 50$, return 1.

(2) If $P_1(b) < 50$, return 0.

(3) For other cases, the embedded bit in the block *b* has been damaged.

4 Conclusion

The 'SysCoP' has been implemented on UNIX platform, and provides a graphical interface, a set (UNIX commands and an API (Application Programming Interface). It currently supports JPEG, PPN GIF, and TIFF image formats. Experiments have been carried out to demonstrate the robustness of ou methods against image processing. For the gray-scaled and color images using the JPEG-base embedding method, three images were labeled, and then processed by JPEG compression, form conversions and color reduction. In general, the results are quite satisfactory and meet the bas requirements for embedding codes as copyright labels. Due to the space limitation of the pape concrete results are omitted. For the binary images, a dithered TIFF binary image and a sharp contrasted TIFF binary image were used in our tests. They are labeled first with R_1= 55% and th robustness degree (λ) 5%. The labeled TIFF images are then smoothed and converted to PBM. Th embedded codes were not damaged in both labeled images after smoothing and conversions.

Our methods are still weak against physical damages (e.g. cut a pixel line, grab an area, etc Currently, we address this problem by allowing the user to specify 'valuable or sensitive' areas of :

image into which labels are (repeatedly) embedded. Thus, cutting a part which is not in these areas does not damage embedded labels.

The methods described in this paper for embedding robust copyright labels for images have been extended to support MPEG-1. Two additional attacks in embedding copyright labels into MPEG-1 videos have been identified: removal of frames and re-compression with different patterns. To be resistant against them, the copyright label is repeatedly embedded into each frame. Thus we ensure that the label can be retrieved from each I-frame regardless of re-compression with different patterns. Furthermore, we are developing new labeling methods for other digital media, i.e. structured electronic documents (e.g. PostScript, SGML documents) and audio data. In addition, a WWW (World Wide Web) image copyright labeling server incorporating the methods described in this paper is being developed.

Acknowledge We are grateful to Scott Burgett from GMI, USA, who initiated and completed the JPEG-based embedding method of 'SysCoP' during his visiting stay at the Fraunhofer-IGD in Darmstadt. We also want to thank Martin Claviez and Joachim Krumb for helping us in implementing the 'SysCoP' system.

References

[1] J. BRASSIL, S. LOW, N. MAXEMCHUK, L. O'GORMAN. Electronic Marking and Identification Techniques to Discourage Document Copying. AT&T Bell Laboratories, Murray Hill, NJ, 1994.

[2] S. BURGETT, E. KOCH, J. ZHAO. A Novel Method for Copyright Labeling Digitized Image Data. Technical Report of Fraunhofer Institute for Computer Graphics, Darmstadt, August, 1994. (Also submitted to IEEE Trans. on Communication, September, 1994).

[3] A.K. CHOUDHURY, N.F. MAXEMCHUK, S. PAUL, H.G. SCHULZRINNE. Copyright Protection for Electronic Publishing over Computer Networks. AT&T Bell Laboratories, June 1994.

[4] W. DIFFIE and M. HELLMAN. New directions in cryptography. *IEEE Transactions on Information Theory*, vol. IT-22, pp. 644-654, 1976.

[5] R. C. DIXON. *Spread Spectrum Systems*, 2nd ed., Wiley, New York, NY, 1984.

[6] B. KAHIN. The strategic environment for protecting multimedia. *IMA Intellectual Property Project Proceedings*, vol. 1, no.1, 1994, pp.1-8.

[7] E. KOCH, J. RINDFREY, J. ZHAO. Copyright Protection for Multimedia Data. *Proceedings of the International Conference on Digital Media and Electronic Publishing* (6-8 December 1994, Leeds, UK).

[8] K. MANTUSI and K. TANAKA. Video-Steganography: How to secretly embed a signature in a picture. *IMA Intellectual Property Project Proceedings*, vol. 1, no. 1, 1994.

[9] W. NIBLACK, R. BARBER, W. EQUITZ, M. FLICKNER, E. GLASMAN, D. PETKOVIC, P. YANKER, C. FALOUTOS, G. TAUBIN. The QBIC Project: Querying Images By Content Using Color, Texture, and Shape. *SPIE* vol. 1908, 1993, pp.173-187.

[10] K.R. RAO and P. YIP. *Discrete Cosine Transform: Algorithms Advantages, Applications*. Academic Press, 1990.

[11] G.K. WALLACE. The JPEG still picture compression standard. *Communications of the ACM*, vol. 34, no. 4, April 1991, pp.30-40.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

HANDBOOK of APPLIED CRYPTOGRAPHY

Alfred J. Menezes
Paul C. van Oorschot
Scott A. Vanstone

CRC Press
Boca Raton London New York Washington, D.C.

Library of Congress Cataloging-in-Publication Data

Menezes, A. J. (Alfred J.), 1965–
Handbook of applied cryptography / Alfred Menezes, Paul van Oorschot, Scott Vanstone.
p. cm. – (CRC Press series on discrete mathematics and its applications)
Includes bibliographical references and index.
ISBN 0-8493-8523-7 (alk. paper)
1. Computers—Access control—Handbooks, manuals, etc.
2. Cryptography--Handbooks, manuals, etc. I. Van Oorschot, Paul C.
II. Vanstone, Scott A. III. Title. IV. Series: Discrete mathematics and its applications.
QA76.9.A25M463 1996
0005.8'2--dc21 96-27609
CIP

Direct all inquiries to CRC Press LLC, 2000 N.W. Corporate Blvd., Boca Raton, Florida 33431.

Visit the CRC Press Web site at www.crcpress.com

International Standard Book Number 0-8493-8523-7
Library of Congress Card Number 96-27609
Printed in the United States of America 5 6 7 8 9 0
Printed on acid-free paper

5.15 Algorithm FIPS 186 one-way function using SHA-1

INPUT: a 160-bit string t and a b-bit string c, $160 \le b \le 512$.
OUTPUT: a 160-bit string denoted $G(t, c)$.

1. Break up t into five 32-bit blocks: $t = H_1 \| H_2 \| H_3 \| H_4 \| H_5$.
2. Pad c with 0's to obtain a 512-bit message block: $X \leftarrow c \| 0^{512-b}$.
3. Divide X into 16 32-bit words: $x_0 x_1 \dots x_{15}$, and set $m \leftarrow 1$.
4. Execute step 4 of SHA-1 (Algorithm 9.53). (This alters the H_i's.)
5. The output is the concatenation: $G(t, c) = H_1 \| H_2 \| H_3 \| H_4 \| H_5$.

5.16 Algorithm FIPS 186 one-way function using DES

INPUT: two 160-bit strings t and c.
OUTPUT: a 160-bit string denoted $G(t, c)$.

1. Break up t into five 32-bit blocks: $t = t_0 \| t_1 \| t_2 \| t_3 \| t_4$.
2. Break up c into five 32-bit blocks: $c = c_0 \| c_1 \| c_2 \| c_3 \| c_4$.
3. For i from 0 to 4 do the following: $x_i \leftarrow t_i \oplus c_i$.
4. For i from 0 to 4 do the following:
   - 4.1 $b_1 \leftarrow c_{(i+4) \bmod 5}$, $b_2 \leftarrow c_{(i+3) \bmod 5}$.
   - 4.2 $a_1 \leftarrow x_i$, $a_2 \leftarrow x_{(i+1) \bmod 5} \oplus x_{(i+4) \bmod 5}$.
   - 4.3 $A \leftarrow a_1 \| a_2$, $B \leftarrow b_1' \| b_2$, where b_1' denotes the 24 least significant bits of b_1.
   - 4.4 Use DES with key B to encrypt A: $y_i \leftarrow \mathrm{DES}_B(A)$.
   - 4.5 Break up y_i into two 32-bit blocks: $y_i = L_i \| R_i$.
5. For i from 0 to 4 do the following: $z_i \leftarrow L_i \oplus R_{(i+2) \bmod 5} \oplus L_{(i+3) \bmod 5}$.
6. The output is the concatenation: $G(t, c) = z_0 \| z_1 \| z_2 \| z_3 \| z_4$.

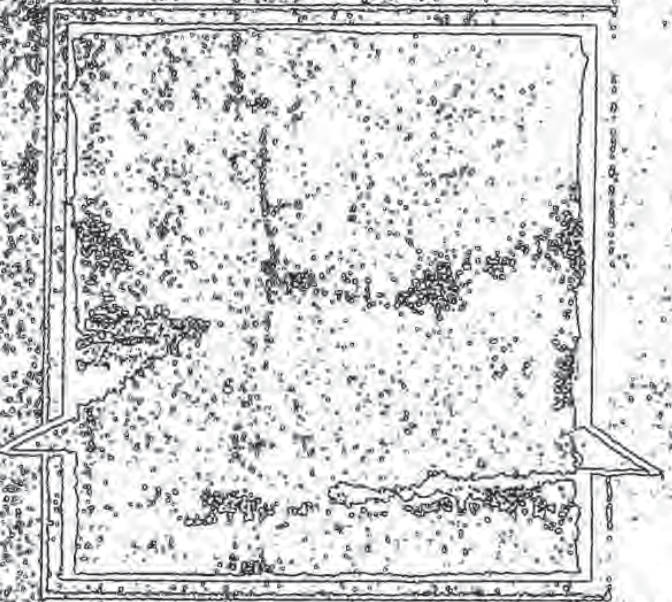
5.4 Statistical tests

This section presents some tests designed to measure the quality of a generator purported to be a random bit generator (Definition 5.1). While it is impossible to give a mathematical proof that a generator is indeed a random bit generator, the tests described here help detect certain kinds of weaknesses the generator may have. This is accomplished by taking a sample output sequence of the generator and subjecting it to various statistical tests. Each statistical test determines whether the sequence possesses a certain attribute that a truly random sequence would be likely to exhibit; the conclusion of each test is not definite, but rather *probabilistic*. An example of such an attribute is that the sequence should have roughly the same number of 0's as 1's. If the sequence is deemed to have failed any one of the statistical tests, the generator may be *rejected* as being non-random; alternatively, the generator may be subjected to further testing. On the other hand, if the sequence passes all of the statistical tests, the generator is *accepted* as being random. More precisely, the term "accepted" should be replaced by "not rejected", since passing the tests merely provides probabilistic evidence that the generator produces sequences which have certain characteristics of random sequences.

§5.4.1 and §5.4.2 provide some relevant background in statistics. §5.4.3 establishes some notation and lists Golomb's randomness postulates. Specific statistical tests for randomness are described in §5.4.4 and §5.4.5.

BEST AVAILABLE COPY

APPLIED CRYPTOGRAPHY



Protocols, Algorithms, and Source Code in C

BRUCE SCHNEIER

Associate Publisher: Katherine Schowalter
Editor: Paul Farrell
Managing Editor: Beth Austin
Editorial Production & Design: Editorial Services of New England, Inc.

This text is printed on acid-free paper.

Library of Congress Cataloging-in-Publication Data
Schneier, Bruce
Applied cryptography : protocols, algorithms, and source code in C / Bruce Schneier.
p. cm.
Includes bibliographical references and index.
ISBN 0-471-59756-2 (paper)
1. Computer security. 2. Telecommunication—security measures. 3. Cryptography. 4. Title

QA76.9.A25S35 1993
005.8'2—dc20

93-2139
CIP

Printed in the United States of America
10 9 8 7 6 5 4 3

Applications of Subliminal Channel

The most obvious application of the subliminal channel is in a spy network. If everyone is sending and receiving signed messages, spies will not be noticed sending subliminal messages in signed documents. Of course, the enemy's spies can do the same thing.

Using a subliminal channel, Alice could safely sign a document under threat. She would, when signing the document, embed the subliminal message, saying, "I am being coerced." Other applications are more subtle. A company can sign documents and embed subliminal messages, allowing them to be tracked throughout the document's lifespan. The government can "mark" digital currency. A malicious signature program can leak the private key. The possibilities are endless.

Subliminal-Free Signatures

Alice and Bob are sending signed messages to each other, negotiating the terms of a contract. They use a digital signature protocol. However, this contract negotiation has been set up as a cover for Alice's and Bob's spying activities. When they use the digital signature algorithm, they don't care about the messages they are signing. They are using a subliminal channel in the signatures to send secret information to each other. The counterespionage service, however, doesn't know that the contract negotiations and the use of signed messages are just cover-ups.

The use of subliminal channels has led people to create **subliminal-free signature schemes.** These are digital signature schemes that cannot be modified to contain a subliminal channel. See [283,284].

4.2 UNDENIABLE DIGITAL SIGNATURES

The Alice Software Company distributes DEW (Do-Everything-Word™). To ensure that their software is virus-free, they include a digital signature. However, they want only legitimate buyers of the software, not pirates, to be able to verify the signature. At the same time, if copies of DEW are found containing a virus, there should be no way for the Alice Software Company to deny a valid signature.

Conventional digital signatures can be copied exactly. Sometimes this property is useful, as in the dissemination of public announcements. Other times they could be a problem. Imagine a digitally signed personal or business letter. If many copies of that document were floating around, each of which could be verified by anyone, this could lead to embarrassment or blackmail. The best solution is a digital signature that can be proven to be valid, but one that the recipient cannot show to a third party without the signer's consent.

Undeniable signatures, invented by David Chaum [207], are suited to these tasks. Like a normal digital signature, an undeniable signature depends on the signed document and the signer's private key. But unlike normal digital signatures, an undeniable signature cannot be verified without the signer's consent.

The mathematics behind this protocol can be found in Section 16.7, but the basic idea is simple:

Shakespeare, or a newsgroup on the Internet [871,872]. There are no keys involved; this is a restricted algorithm.

Gustavus Simmons invented the concept of a subliminal channel in a conventional digital signature algorithm [821]. Since the subliminal messages are hidden in what looks like normal digital signatures, this is a form of obfuscation. Walter sees signed innocuous messages pass back and forth, but he completely misses the information being sent over the subliminal channel. In fact, the subliminal-channel signature algorithm is indistinguishable from a normal signature algorithm, at least to Walter. Walter not only cannot read the subliminal message, but he also has no idea that one is even present. (Of course, any warden who gives his prisoners computers and high-speed modems deserves what he gets.)

In general the protocol looks like this:

(1) Alice generates an innocuous message, at random.

(2) Using a secret key shared with Bob, Alice signs the innocuous message in such a way as to hide her subliminal message in the signature. (This is the meat of the subliminal channel protocol; see Section 16.6.)

(3) Alice sends this signed message to Bob via Walter.

(4) Walter reads the innocuous message and checks the signature. Finding nothing amiss, he passes the signed message to Bob.

(5) Bob checks the signature on the innocuous message, confirming that the message came from Alice.

(6) Bob ignores the innocuous message and, using the secret key he shares with Alice, extracts the subliminal message.

What about cheating? Walter doesn't trust anyone and no one trusts him. He can always prevent communication, but he has no way of introducing phony messages. Since he can't generate any valid signatures, Bob will detect his attempt in step (3). And since he does not know the shared key, he can't read the subliminal messages. Even more important, he has no idea that the subliminal messages are there. Signed messages using a digital signature algorithm look no different from signed messages with subliminal messages embedded in the signature.

Cheating between Alice and Bob is more problematic. In some implementations of a subliminal channel, the secret information Bob needs to read the subliminal message is the same information Alice needs to sign the innocuous message. If this is the case, Bob can impersonate Alice. He can sign messages purporting to come from her, and there is nothing Alice can do about it. If she is to send him subliminal messages, she has to trust him not to abuse her private key.

Other subliminal channel implementations don't have this problem. A secret key shared by Alice and Bob allows Alice to send Bob subliminal messages, but it is not the same as Alice's private key and does not allow Bob to sign messages. Alice does not have to trust Bob not to abuse her private key.

BEST AVAILABLE COPY

DIGITAL AUDIO CARRYING EXTRA INFORMATION

W.R.Th. ten Kate, L.M. van de Kerkhof and F.F.M. Zijderveld

Philips Research Laboratories
P.O.Box 80000
5600 JA Eindhoven, The Netherlands

ABSTRACT

A new technique is proposed which enables the inaudible addition of extra information to an audio signal. Full compatibility with present-day transmission and reproduction systems is guaranteed, as the resulting signal remains in the same format. When reproducing the new signal on such equipment there will be no perceptible difference. The extra information, however, can be retrieved by the use of additional signal processing. The technique of adding and retrieving extra information is described and applications are presented. Special attention is given to the so-called 4-2-4 coding, as the technique presented offers a very promising scheme for such a coding.

INTRODUCTION

The proposed technique is based on the masking effect [1]. Masking is the psycho-acoustic phenomenon which deals with the insensitivity of the human ear to sounds in the presence of other, louder sounds. It is usually explained in terms of an upward shift in the hearing threshold, which is then referred to as the masking threshold. Masking is most effective for frequencies close to the frequency of the masking sound, but extends to both lower and higher frequencies, diminishing more rapidly to the lower than to the higher end. The same holds for the time characteristics: masking is strongest for sounds occurring simultaneously, but is also observed in the time spans shortly before and after the presentation of the masking sound.

A direct consequence of the masking effect is that it allows the inaudible addition of extra information to an audio signal. The inaudibility is guaranteed if the sound power level of the added signal is kept below the masking threshold.

In order to obtain a system carrying out such an addition, a method is required that adapts the extra information to the frequency and time structure of the audio signal in such a way that the masking rules known from psycho-acoustics are satisfied. Of course, the addition must be performed in such a way that the retrieval of the extra information is also possible. This paper proposes such a method.

The techniques on which the method is based are quite similar to those used in subband coding systems [2], see Fig.1. The signals are filtered by a filter bank into subband signals, on which the addition operation is subsequently performed. After addition, the resulting subband samples are synthesized to a broadband signal. This signal will be perceived as the original audio signal. The extra information is retrieved by applying the retrieval operation to the subband signals, which have first been obtained by splitting the broadband signal.

The principle behind the addition and retrieval operation is the quantization of the (subband) samples. It is the quantizing that makes it possible to add another signal in such a way that it can also be recovered: by applying the same quantizing to the signal which has resulted after the addition, the extra information is identified as the "quantizing noise".

The resulting broadband signal can be represented in formats according to present standards. So full compatibility is guaranteed and the signal can be transmitted and reproduced by the conventional systems. There will be no perceptible difference from the original audio signal if the power of the error signal is kept below the masking threshold. The error signal consists of two main components, namely the added signal and the error signal which is solely due to the applied quantization.

In order to receive the extra information as well, extra processing is required. This processing consists of filtering the broadband signal into subband signals again, quantizing these subband signals and specifying the quantization noise, which represents the information required.

In the next sections the method will be outlined in more detail. Subsequently, some applications will be presented. The concluding section summarizes the presented ideas.

ADDITION AND RETRIEVAL

The masking audio signal will be referred to as the main signal (M), the information to be added as the auxiliary signal (A). In order to clarify the addition and retrieval mechanisms, it is assumed that the auxiliary signal is itself an audio signal. Moreover, the auxiliary and main signal are assumed

CH2847-2/90/0000-1097 $1.00 © 1990 IEEE

to exhibit some correlation. Based on these assumptions the method can be explained as follows:

Because the masking properties of a signal are dependent on both its time and frequency structure, the audio signals *M* and *A* are first filtered into subbands, see Fig.1. To make a suitable choice of the bandwidths of these subbands a trade-off has to be made between benefitting from the masking effect and the resulting technical complexity [2]. The analyzing and synthesizing filters are required to be (nearly) perfect-reconstructing [3].

As in subband coding [2], the power of the main signal in each subband is estimated. Based upon these estimates, the maximal signal level which can be masked in each subband is determined. Next, the subband components of the main signal are quantized according to these calculated levels. The quantizing should be such that the resulting quantizing noise together with the added information will be masked by the main signal. The quantizing operation is described by

$$Q(sample) := qstep * \mathrm{ROUND}(sample/qstep). \qquad (1)$$

where *sample* represents the value of the sample being quantized, *Q(sample)* its value after the quantizing, and *qstep* the step size according to which the sample is quantized. Clearly, if the signal has a large masking capability, a large value for *qstep* can be assigned. In fact, the quantizing is the principal step by which it is possible to add and retrieve information. After quantizing, the main-signal's subband-samples can only be integer multiples of *qstep*. Consequently, the samples' values are allowed to be changed by adding any value in the range $(-\frac{1}{2}qstep, +\frac{1}{2}qstep)$, as they can be recovered by applying the quantizing operation again. What is more, the added value can then be determined as well, as it appears as the difference between the changed and requantized value, see Fig.2. Clearly, this value is going to represent the auxiliary signal.

The resulting subband signals are synthesized to a broadband signal prior to transmission. Because of its compatibility it can be reproduced by the equipment currently in use. There will be no perceptible difference. The extra information can be retrieved by adding extra processing circuitry to this equipment. This extra processing consists of filtering the signal into subband signals again and determining the added values by applying the quantizing operation Eq.(1). For the sake of clarity it should be noted that the "transmit" action in Fig.2 also represents the synthesizing and analyzing filtering.

The quantizing can be interpreted as the addition of a noise signal with a power given by [4]

$$\frac{qstep^2}{12} \qquad (2)$$

Similarly, the addition representing the auxiliary signal contributes to the noise in the main signal. It is reasonable to suppose that the upper bound of the power of this addition is also given by Eq.(2). As the addition and quantization noise are uncorrelated, a total power of at most twice the value expressed by Eq.(2) results as the noise power to be masked by the main signal.

The addition of the auxiliary signal to the main signal is to be performed by adding numbers with a value in the range $(-\frac{1}{2}qstep, +\frac{1}{2}qstep)$. These numbers are obtained by attenuating the auxiliary signal sufficiently, c.f. Fig.2. The amount of attenuation depends on the strength of the auxiliary signal and on the available room *qstep*. As the main and auxiliary signal are correlated these two parameters are related as well, and the required attenuation factor can be obtained as follows.

When *L* levels are used to represent the main signal, *L* being an odd integer, the maximum absolute value of the main signal after quantization will be

$$\frac{L-1}{2}\,qstep. \qquad (3A)$$

So the largest possible value before quantization is given by

$$\frac{L}{2}\,qstep. \qquad (3B)$$

Assuming that the main and auxiliary signal are fully correlated, the values of the auxiliary-signal samples will also range up to the maximum given by Eq.(3B). Consequently, the attenuation factor to be applied to the auxiliary signal results as

$$gain = \frac{\frac{qstep}{2}}{\frac{L}{2}\,qstep} = \frac{1}{L}. \qquad (4)$$

Because in practice the two signals are not fully correlated, some further attenuation, by a fixed amount of, for example, 0.80, should be applied to avoid quantize overload.

After attenuation it must be checked whether the resulting value is in the required range of a half *qstep*. If not, a clipping to $\frac{1}{2}qstep$ will be performed. Experiments have shown that quite a large number of clippings can be tolerated before they will become perceptible audibly. The reason for this is attributed to the masking properties of the auxiliary signal. Because the errors due to clipping are confined to the same frequency range and because they only occur at the large valued samples, they can be masked by the auxiliary signal.

In order to retrieve the auxiliary signal from a received (broadband) signal, first the filtering into subbands has to be applied. When, after quantizing the subband samples, the sizes of the added numbers are determined, these numbers have to be amplified to obtain the original strength of the auxiliary signal. The amplification factor is given by the inverse of Eq.(4), corrected for the amount of further applied attenuation. Synthesizing the resulting subband signals will finally lead to the retrieval of the auxiliary signal. Fig.3 depicts the total scheme of addition and retrieval in further detail.

DISCUSSION

In the foregoing it was assumed that the auxiliary signal was correlated with the main signal. For this case a scheme for the addition was presented. In the case of uncorrelated signals, however, it is still possible to add (and retrieve) both signals under the constraint that only one of them can be perceived. In that case, we consider the auxiliary signal as an arbitrary bit stream representing some information. (It is, of course, apt to first apply a data-reduction algorithm [2] to the auxiliary signal before adding it to the main signal.)

Depending on the room available at a certain moment, a number of bits can be added to the main signal. The available room is determined by *qstep* (Eq.(1)). The group of bits to be added at that instant are transformed into a number, e.g. '101' is transformed into '5'. Next, the number is scaled so that it fits into the range [0,*qstep*). So, in the given example a scaling by *qstep*/8 is to be applied. This scaling parameter directly indicates how the number of bits which can be added depends on the size of *qstep*. Finally, a shift over -½*qstep* is applied, so that we arrive at a number which is in the range [-½*qstep*, +½*qstep*). The further processing continues as described in the previous section for correlated signals.

In the paper the filtering is assumed to be performed according to a subband scheme, as we believe that such a scheme offers the best results obtainable in a technical implementation. As far as the ideas described are concerned, however, the choice of the actual frequency-selective scheme is of no importance.

Concerning the filter banks there is a point which requires consideration. In order to reduce the system's complexity, multirate filter banks are used. Such filter banks show a time-variant behavior with a period of the largest down sampling factor. Perfect-reconstruction-type filter banks are designed in such a way that this time variance is not noticeable at the input and output terminals. Here, input and output terminals refer to the usual filter-bank configuration, which is the configuration in which the input and output signals are in the broadband format having the highest sample rate. In our case, however, the other configuration is used as well, i.e. the configuration of filter banks which synthesize subband signals at their input to a broadband signal and subsequently split this broadband signal into subband signals again. In fact, the "transmit" action in Fig.2 represents this configuration.

With respect to this new configuration, the filter banks are required to be (nearly) perfect-reconstructing as well. This requirement is automatically met by the perfect-reconstruction condition for the usual configuration. This can be seen by considering the cascade of an analyzing, a synthesizing and again an analyzing filter bank. Because the filters are designed so that the two broadband signals in this cascade are equal, the subband signals at the input of the synthesizing filter bank should be equal to those at the output of the second analyzing filter bank.

The time-variant behavior should also remain unnoticeable in the new filter-bank configuration (i.e. synthesizing followed by analyzing). This is only achieved if we ensure that the total delay is an integer times the sampling period at the lowest sampling rate, or, equivalently, when expressed in sampling periods at the highest sampling rate, if the total delay is an integer times the highest downsampling factor. In other words, it is necessary to synchronize the downsamplers in the analyzing filter bank at the retrieval (receiver) site with the upsamplers in the synthesizing filter bank at the addition (transmitter) site.

On the other hand, when no up- and downsampling is applied, the time-invariance condition will be satisfied, at the expense, however, of automatically fulfilling the perfect reconstruction demand. Because of the bandpass properties of the filters in the filter bank this demand can only be satisfied when the subband signals are sampled at their critical rate, because in that case the bandwidth of the bandpass filter matches that of any conceivable subband signal (Remind that due to the processing the spectrum of the subband samples is extended over the total available bandwidth given by the sampling rate).

APPLICATIONS

The applications of this new technique are manyfold. It is convenient to make a classification of the information to be added according to its representation; if it is another (correlated) audio signal or if it is simply representing data (bits). Different processing strategies are required in each case. Audio signals are to be divided into correlated and uncorrelated signals. As mentioned in the previous section, uncorrelated auxiliary signals are to be considered as a bit stream of data and processed accordingly. The processing of correlated signals was described in the main section of the paper. The data can be, for example, extra features (control data). Examples of correlated signals are an anechoic recording and a dummy-head recording.

A very convincing application of this new technique concerns the so-called 4-2-4 coding. Nowadays movies often have four audio channels. Up to now for use at home in TV sets the signal has been constrained to two channels, as the room available for transmission does not allow more. For this reason the original four signals are mixed into a stereo signal, thereby nullifying the 4-channel sound sensation. To overcome this problem, the mixing is usually performed by some kind of matrixing by which it is possible to retrieve some impression of the original four channels [5].

With the technique presented, however, it is now possible to generate a 2-channel signal which is fully compatible with the mixed stereo signal, but which can also be converted into the original 4-channel version. The following matrix can be used to mix the original four signals L, R, C and S:

$$M_1 = L + ½\sqrt{2}\ (C + S), \quad (5A)$$

$$M_2 = R + ½\sqrt{2}\ (C + S), \quad (5B)$$

$$A_1 = ½\sqrt{2}\ (C + S), \quad (5C)$$

$$A_2 = ½\sqrt{2}\ (C - S), \quad (5D)$$

The signals M_1 and M_2 compose the new left and right stereo signals respectively. In addition, they each serve as a main signal to carry one of the (auxiliary) signals A_1 or A_2. As expressed by Eqs.(5), the main and auxiliary signals are correlated, and consequently the described addition and retrieval algorithms for such signals can be applied.

Listening to the resulting signals produces the sensation that the pure M_1 and M_2 are being transmitted. By applying the necessary additional processing, the auxiliary signals A_1 and A_2 can be retrieved. By subsequently executing the inverse matrix operation of Eqs.(5) we will finally arrive at four signals which exhibit the same perceptual experience as the original four. Note that, besides the much better channel resolution obtainable with this coding scheme, the composition of the compatible stereo signal out of the four original signals is done in a much more attractive way than in other 4-2-4 coding schemes [5,6].

The performance of a real-time version of such a 4-2-4 coding system confirmed our expectations, demonstrating its power as well as the improvement obtainable compared with the mixing procedures in use until now.

SUMMARY

A new technique, based on the masking effect, is proposed which enables the inaudible addition of extra information to an audio signal. It opens the way to a whole new area of extensions for audio and/or video with audio, while remaining compatible with present-day standards.

As a major application of this new technique a new 4-2-4 coding scheme is presented, having an increased channel resolution. The scheme offers full compatibility with present 2-channel transmission and reproduction systems, while through the incorporation of extra processing it also provides the possibility of reproducing the original 4-channel sound sensation.

ACKNOWLEDGEMENT

The authors would like to express their gratitude to Dr. W.F. Druyvesteyn, who first came up with the idea of information addition, and to Dr. R.N.J. Veldhuis, who offered the basic algorithms for its realization. His careful reading of the manuscript was of great assistance and was greatly appreciated.

REFERENCES

[1] E. Zwicker, R. Feldtkeller, *Das Ohr als Nachrichtenempfänger*, S. Hirzel Verlag, Stuttgart, 1967

[2] R.N.J. Veldhuis, M. Breeuwer and R. van der Waal, *Subband coding of digital audio signals*, Philips J. Res. 44 (1989) 329-343

[3] M. Vetterli and D. LeGall, *Perfect reconstruction FIR filter banks: some properties and factorizations*, IEEE Trans. ASSP 37 (1989) 1057-1071

[4] N.S. Jayant and P. Noll, *Digital coding of waveforms*, Prentice-Hall, Englewood Cliffs, NJ, 1984

[5] S. Julstrom, *A high-performance surround sound process for home video*, J. Audio Eng. Soc. 35 (1987) 536-549

[6] J.M. Eargle, *Multichannel stereo matrix systems: an overview*, J. Audio Eng. Soc. 19 (1971) 552-559

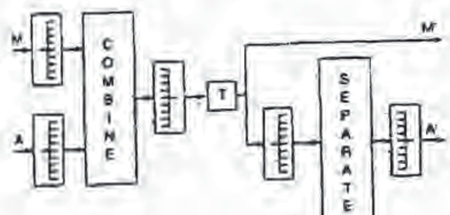


Fig.1. General block diagram

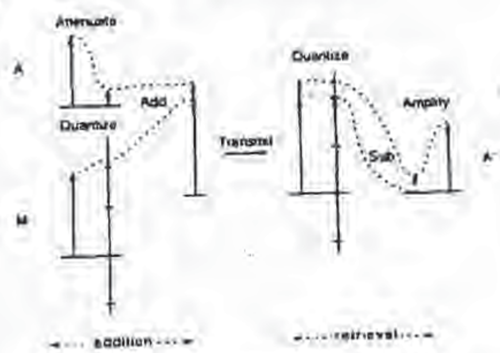


Fig.2. The process of "addition" and "retrieval"

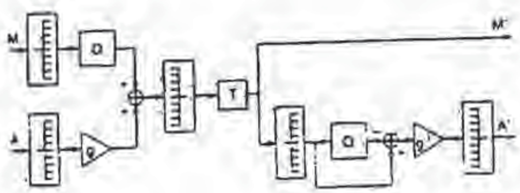


Fig.3. Total scheme

PROCEEDINGS

ICIP-94

Volume II of III

November 13 - 16, 1994
Austin Convention Center
Austin, Texas

Sponsored by
The Institute of Electrical and Electronics Engineers Signal Processing Society

IEEE

IEEE Computer Society Press
Los Alamitos, California
Washington • Brussels • Tokyo

IEEE Computer Society Press
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1264

IEEE Computer Society Press Order Number 6950-02
Library of Congress Number 94-78171
IEEE Catalog Number 94CH35708
ISBN 0-8186-6950-0 (paper)
ISBN 0-8186-6951-9 (microfiche)
ISBN 0-8186-6952-7 (case)

Additional copies may be ordered from:

IEEE Computer Society Press
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1264
Tel. +1-714-821-8380
Fax: +1-714-821-4641
Email: cs.books@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: +1-908-981-1393
Fax: +1-908-981-9667

IEEE Computer Society
13, Avenue de l'Aquilon
B-1200 Brussels
BELGIUM
Tel. +32-2-770-2198
Fax: +32-2-770-8505

IEEE Computer Society
Ooshima Building
2-19-1 Minami-Aoyama
Minato-ku, Tokyo 107
JAPAN
Tel: +81-3-3408-3118
Fax: +81-3-3408-3553

Editorial production by Bob Werner
Printed in the United States of America by Braun-Brumfield, Inc.

The Institute of Electrical and Electronics Engineers, Inc.

A DIGITAL WATERMARK

R.G.van Schyndel(*), A.Z.Tirkel(+), C.F.Osborne(*)

(*) Department of Physics, Monash University, Clayton, 3168, Australia.
(+) Scientific Technology, 21 Walstab St, E. Brighton, 3187, Australia.

ABSTRACT

This paper discusses the feasibility of coding an "undetectable" digital water mark on a standard 512*512 intensity image with an 8 bit gray scale. The watermark is capable of carrying such information as authentication or authorisation codes, or a legend essential for image interpretation. This capability is envisaged to find application in image tagging, copyright enforcement, counterfeit protection, and controlled access. Two methods of implementation are discussed. The first is based on bit plane manipulation of the LSB, which offers easy and rapid decoding. The second method utilises linear addition of the water mark to the image data, and is more difficult to decode, offering inherent security. This linearity property also allows some image processing, such as averaging, to take place on the image, without corrupting the water mark beyond recovery. Either method is potentially compatible with JPEG and MPEG processing.

1. INTRODUCTION

The art/science of hiding messages is known as steganography [1]. Conventional techniques involve the encryption of a copyright message on one colour of a composite image. The method described in this paper relies on the manipulation of the LSB of any *colour or monochrome* image, in a manner which is undetectable to the eye. The embedded message is decoded and can removed from this modified image in order to recover the original information. The desirable properties of an electronic water mark are undetectability and accurate recovery of the hidden message. In general, the problem of embedding an invisible watermark and its subsequent extraction falls into the category of matched or adaptive filtering [2]. The authors have developed a simple modification of such a scheme. In order to render the watermark undetectable, encoding with m-sequences was chosen, because of their balance, random appearance and good auto-correlation properties (a single peak with no sidelobes), which simplify the recovery process [3]. In practice, extended m-sequences were employed, being commensurate with the image size (2^n) and exhibiting a null in autocorrelation around the main peak [4]. Two dimensional analogues such as Costas Arrays were studied, but were rejected because of their sparse nature [5]. For simplicity, we have chosen to encode the water mark by the choice of m-sequence phase. (An alternative method could use the choice of m-sequence to determine the data byte). This paper demonstrates the feasibility of such encoding and the accuracy of the message extraction.

2. M-SEQUENCES

M-Sequences can be formed from starting vectors by a Fibonacci recursion relation, which can be implemented by linear shift registers. They are of maximal length (2^n-1) for a vector of length n. The sequences thus formed (or the polynomials by which they can be generated) form a finite field called Galois Field. The autocorrelation function and spectral distribution of m-sequences resemble that of random Gaussian noise. The cross-correlation of m-sequences has been examined mathematically and empirically in [6], [7] and [8]. Certain families of sequences (maximal connected sets) have been known to possess desirable cross-correlation properties [9]. Images encoded with m-sequences or one bit Gaussian noise are statistically indistinguishable from each other and only visually distinguishable from the original if the image contains large areas with a small intensity variation. In many imaging systems the LSB is corrupted by hardware imperfections or quantisation noise and hence its sacrifice is of limited significance. The exact choice of code depends on the amount of data to be embedded, the errors involved in image transmission, and the degree of security required [10]. The Monash group has performed extensive analysis of m-sequence codes and their correlations [8]. The vulnerability of m-sequences to cracking is characterised by their span (2n), which is the dimension of the matrix which must be diagonalised in order to determine the shift register configuration [3]. In the case of the linear addition of the m-sequence to the image LSB, the code cracker must know the image content without errors in order to determine the encoding sequence. The span of these sequences can be increased by forming compound codes (Gold or Kasami) or by performing non-linear mappings, such as in the GMW sequences [3]. The number of available sequences varies according to the operations performed. Also, it is possible to utilise other sequences of the de Bruijn type, such as Legendre sequences, based on residues, and extremely difficult to crack [11].

3. METHODS OF INCORPORATING THE WATER MARK

Our experiments were conducted on 512*512 8 bit gray scale images encoded on a line by line basis with m-sequences 2^n pixels long. The first method involves the embedding of the m-sequence on the LSB of the image data. The original 8 bit gray scale image data is capable of compression to 7 bits by adaptive histogram manipulation. this process is followed by a compensating mapping

0-8186-6950-0/94 $4.00 © 1994 IEEE

restore the dynamic range, the resulting image is practically indistinguishable from the original. The above process enables the LSB to carry the watermark information. The watermark can be decoded by comparing the LSB bit pattern with a stored counterpart. The watermark message can be carried by the choice of sequence (or its complement) and its phasing. A schematic equivalent of the decoder is shown in Fig.1.

The second method uses LSB addition for embedding the water mark. As a result, the decoder is more complex, as shown in Fig.2. The decoding process makes use of the unique and optimal auto-correlation function of m-sequences. The process requires the examination of the complete bit pattern, and in its current implementation, must therefore be performed off-line, which is its principal disadvantage. However, it is intrinsically more secure, since a potential code breaker has to perform the same operations, without any a-priori knowledge. The decoding process is not completely error free, due to partial correlation of the image data with the encoding sequence. This may be suppressed by a deliberate compression of the image dynamic range followed by a compensating mapping of the lookup table, which leads to gray scale quantisation effects. Analysis of the image histogram indicates that a 3 bit dynamic range compression (from 8 bits down to 5 bits) should permit threshold detection to be successful. Alternatively the application of a longer sequence of length 2^{15} or better filtering or detection algorithms should have a similar effect. Since the auto-correlation peak is typically very sharp, (superimposed on a significant, but slowly varying background) it is possible to employ simple filtering techniques to extract it. Various length impulse response filters were tested in single and multi-pass configurations. The differential filter (with kernel -1,-1,4,-1,-1) was found to yield optimum results for 512*512 images with a single M-sequence. The separation of the image crosscorrelation histogram into image and message peaks shown in Fig. 3(a) and (b) demonstrate the feasibility of using thresholding on the cross-correlation values as a simple and rapid technique of message decoding. The possibility of false positives and false negatives was also investigated in terms of the effect the image content has on the auto-correlation function. The effects of adding two distinct messages to the same image each encoded using a different m-sequence and then added to the image was investigated in terms of their effect on each other and their recovery ability. Such an image could contain two watermarks, one for the hospital, and one for the radiologist. Fig 4. shows some images to which the water mark has been added. The composite image of Fig 4(a) has been changed to that of Fig 4(b) with the addition of the water mark, with a enlarged detail shown in 4(c). The cross-correlation after filtering of this image is shown with (above) and without (below) the watermark in Fig 4(d). The peaks are visible as a series of single white dots in the lower right portion of the figure, whose position determines the message letter (in this case the message is "aabbccAABBCC " repeated four times). This suggests that the water mark is undetectable only in circumstances where low-level gaussian noise is expected. Typically this does not include computer generated images.

4 APPLICATIONS

The objectives of this project are to investigate the feasibility of embedding undetectable watermarks for the purposes of image integrity verification, tagging and copyright infringement protection and controlled image access. The anticipated applications include medical images, commercial photographs and videos, sensitive documents such as patents, artwork, and computer generated images.

5. FUTURE WORK

The authors are investigating LMS adaptive filter extraction algorithms to determine an optimum technique with minimal image dependence. We will explore the effects on the watermark due to cropping and distortions such as skew, rotations, translations etc. and countermeasures against these. These may include bit-swapping or diagonal raster folding of sequences into m-arrays [12]. These operations are facilitated by our choice of extended m-sequences of length 2^n. The desirability of tamper-resistant watermarks could be a function of the application. Some implementations may be better served by retaining a distorted watermark as evidence of the illegal act! This aspect requires further study.

6. CONCLUSIONS

This paper examines the feasibility of embedding a digital water mark on test images. The main problems found with adding the water mark is in retaining the dynamic range of the original image and the auto-correlation output. The paper discusses a method which would avoid the sacrifice of the LSB for the insertion of the sequence, and the ramifications on image processing and compression. The techniques used and contemplated for watermark coding and detection are all compatible with hardware implementation in standard size programmable gate array IC's. Such implementation would be capable of on-line, real time algorithm execution.

7.ACKNOWLEDGEMENT

The authors would like to express their gratitude to Mr. G.A.Rankin for his assistance in developing a program to generate and analyse the auto and cross-correlations of m-sequences and related codes, which has proved invaluable in this project.

8. REFERENCES

[1] E.Sapwater and K.Wood "Electronic Copyright Protection", Photo-Electronic Imaging, vol 37, No.6, (1994), p.16-21.

[2] B.Widrow. "Adaptive Signal Processing," Englewood Cliffs, N.J. Prentice Hall, 1985.

[3] M.K. Simon, J.K.Omura, R.A.Scholtz, B.K.Levitt. "Spread Spectrum Communications" Volume III. Rockville Md. Computer Science Press. 1985.

K 4191

[4] U.-C.G.Fiebig "Auto- and Crosscorrelation Properties for Extended m-Sequences and Related Sequences" IEEE ISSTA Symposium, Oulu, Finland, July 4-6, 1994, p.406-410.
[5] S.W.Golomb, H.Taylor "Construction and Properties of Costas Arrays". Proc.IEEE, vol.72, p.1143-1163, Sept 1984.
[6] Sarwate D.V., Pursley M.B. " Crosscorrelation Properties of Pseudorandom and Related Sequences" Proc. of the IEEE vol.68, no 5, May 1980, pp 593-619.
[7] Niho Y. " Multi-valued Cross-correlation Functions Between Two Maximal Linear Recursive Sequences" Ph.D. Dissertation, Department of Electrical Engineering, University of Southern California 1972.
[8] A.Z.Tirkel, N.R.A.Mee, C.F.Osborne, G.A.Rankin "Cross-Correlation Properties of M-Sequences" Paper Submitted to IEEE Transactions on Information Theory.
[9] A.Z.Tirkel, C.F.Osborne, N.Mee, G.A.Rankin, A.McAndrew, "Maximal Connected Sets - Application to Microcell CDMA".International Journal of Communication Systems, 1994, vol 7, p.29-32.
[10] A.Z.Tirkel, G.A.Rankin, R.M.van Schyndel, W.J.Ho, N.R.A.Mee, C.F.Osborne "Electronic Water Mark", DICTA-93 Macquarie University, Sydney, December 1993, p.666-672.
[11] S.Kitabayashi, T.Ozawa, M.Hata. "Property of the Legendre Subsequence" Communication on the Move. ICCS/ISITA '92, Singapore, vol 3, p.1224-1228.
[12] F.J.McWilliams and N.J.A.Sloane. "Pseudorandom Sequences and Arrays". Proc.IEEE(76), vol 64, p.1715-1729.

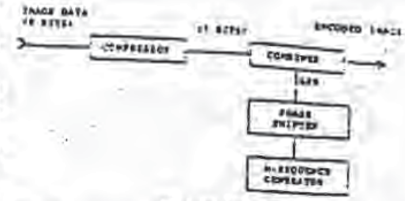


FIG 1 (a) BIT PLANE ENCODER

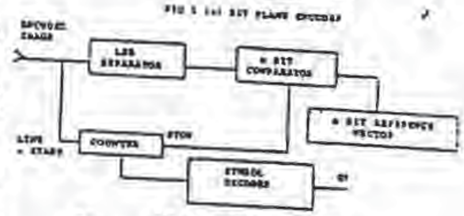


FIG 1 (b) BIT PLANE DECODER

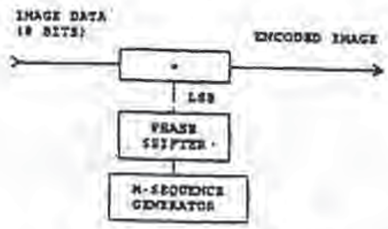


FIG 2 (a) ADDITIVE ENCODER

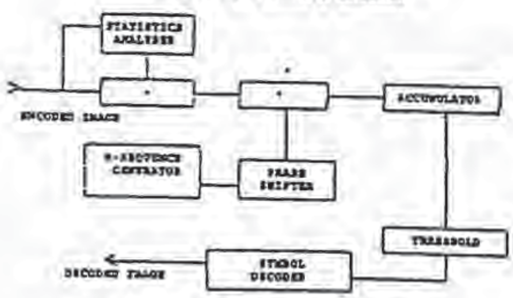


FIG 2 (b) ADDITIVE DECODER

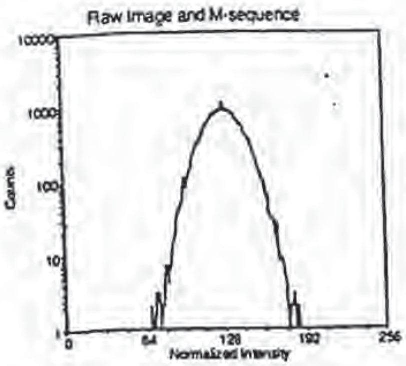


Fig. 3 (a)

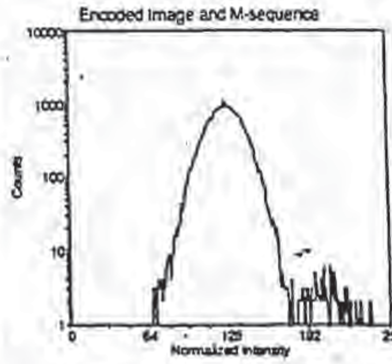


Fig. 3(b)

Fig. 3 Histograms of the Cross-correlation of the Raw and Encoded Images with the Watermark M-Sequence.

Fig. 4 (a) Upper - Unencoded Composite Image
Fig. 4 (b) Lower - Composite Image *with* Watermark (Undetectable)

Fig. 4(c) Detail of Binary Test Image Showing the Watermark (Enhanced Contrast)

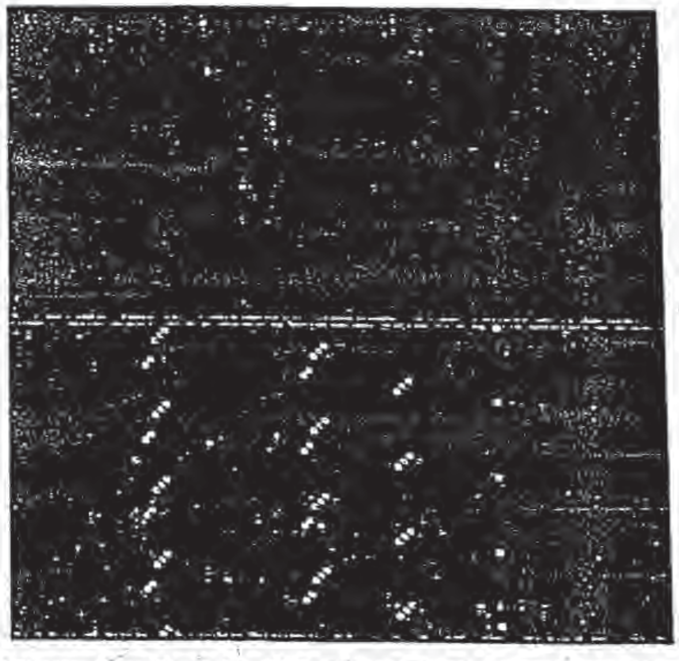


Fig. 4(d) M-Sequence Cross-Correlation with Raw Image (Upper), Encoded Image (Lower)
(Watermark Message Reads "aabbccAABBCC")

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

Springer
Berlin
Heidelberg
New York
Barcelona
Budapest
Hong Kong
London
Milan
Paris
Santa Clara
Singapore
Tokyo

Ross Anderson (Ed.)

Information Hiding

First International Workshop
Cambridge, U.K., May 30 - June 1, 1996
Proceedings

(1)

Springer

12. E. Koch, J. Rindfrey, and J. Zhao. Copyright protection for multimedia data. In *Proc. of the Int. Conf. on Digital Media and Electronic Publishing*, 1994.
13. E. Koch and Z. Zhao. Towards robust and hidden image copyright labeling. In *Proceedings of 1995 IEEE Workshop on Nonlinear Signal and Image Processing*, June 1995.
14. F. T. Leighton and S. Micali. Secret-key agreement without public-key cryptography. In *Proceedings of Crypto*, 1993.
15. J.S Lim. *Two-Dimensional Signal Processing*. Prentice Hall, Englewood Cliffs, N.J., 1990.
16. B. M. Macq and J-J Quisquater. Cryptology for digital tv broadcasting. *Proc. of the IEEE*, 83(6):944–957, 1995.
17. K. Matsui and K. Tanaka. Video-steganography. In *IMA Intellectual Property Project Proceedings*, volume 1, pages 187–206, 1994.
18. R. L. Pickholtz, D. L. Schilling, and L. B. Millstein. Theory of spread spectrum communications - a tutorial. *IEEE Trans. on Communications*, pages 855–884, 1982.
19. W. F. Schreiber, A. E. Lippman, E. H. Adelson, and A. N. Netravali. Receiver-compatible enhanced definition television system. Technical Report 5,010,405, United States Patent, 1991.
20. K. Tanaka, Y. Nakamura, and K. Matsui. Embedding secret information into a dithered multi-level image. In *Proc. 1990 IEEE Military Communications Conference*, pages 216–220, 1990.
21. L. F. Turner. Digital data security system. Patent IPN WO 89/08915, 1989.
22. R. G. van Schyndel, A. Z. Tirkel, and C. F. Osborne. A digital watermark. In *Int. Conf. on Image Processing*, volume 2, pages 86–90. IEEE, 1994.

Modulation and Information Hiding in Images

Joshua R. Smith and Barrett O. Comiskey

{jrs, elwood}@media.mit.edu
Physics and Media Group
MIT Media Lab
20 Ames Street
Cambridge, MA 02139
USA

Abstract. We use concepts from communication theory to characterize information hiding schemes: the amount of information that can be hidden, its perceptibility, and its robustness to removal can be modeled using the quantities channel capacity, signal-to-noise ratio, and jamming margin. We then introduce new information hiding schemes whose parameters can easily be adjusted to trade off capacity, imperceptibility, and robustness as required in the application. The theory indicates the most aggressive feasible parameter settings. We also introduce a technique called *predistortion* for increasing resistance to JPEG compression. Analogous tactics are presumably possible whenever a model of anticipated distortion is available.

1 Introduction

In this paper, we discuss schemes for imperceptibly encoding extra information in an image by making small modifications to large numbers of its pixels. Potential applications include copyright protection, embedded or "in band" captioning and indexing, and secret communication.

Ideally, one would like to find a representation that satisfies the conflicting goals of not being perceivable, and being difficult to remove, accidentally or otherwise. But because these goals *do* conflict, because it is *not* possible to simultaneously maximize robustness and imperceptibility, we will introduce a framework for quantifying the tradeoffs among three conflicting figures of merit useful for characterizing information hiding schemes: (1) capacity (the number of bits that may be hidden and then recovered) (2) robustness to accidental removal, and (3) imperceptibility. We will then present new information hiding schemes that can be tailored to trade off these figures of merit as needed in the particular application. For example, capacity may be more important in a captioning application, robustness may be most desired for copyright protection schemes, and imperceptibility might be favored in a secret communication scenario.

1.1 Information theoretic view of the problem

We view an image in which extra information has been embedded as an approximately continuous (in amplitude), two-dimensional, band-limited channel with large average noise power. The noise is the original unmodified image, which we will refer to as the *cover image*, and the signal is the set of small modifications introduced by the hider. The modifications encode the *embedded message*. We will refer to the modified, distribution image as the *stego-image*, following the convention suggested at the Information Hiding Workshop. From this point of view, any scheme for communicating over a continuous channel—that is, any modulation scheme—is a potential information hiding scheme, and concepts used to analyze these schemes, such as channel capacity, ratio of signal power to noise power, and jamming margin can be invoked to quantify the trade-offs between the amount of information that can be hidden, the visibility of that information, and its robustness to removal.

1.2 Relationship to other approaches

In our framework, it becomes obvious why *cover image escrow* hiding schemes such as those presented in [CKLS] and [BOD95] have high robustness to distortion. In cover image escrow schemes, the extractor is required to have the original unmodified cover image, so that the original cover image can be subtracted from the stego-image before extraction of the embedded message. Because the cover image is subtracted off before decoding, there is no noise due to the cover image itself; the only noise that must be resisted is the noise introduced by distortion such as compression, printing, and scanning. While the image escrow hiding schemes must respect the same information theoretic limits as ours, the noise in their case is very small, since it arises solely from distortions to the stego-image.

In our view, image escrow schemes are of limited interest because of their narrow range of practical applications. Since the embedded message can only be extracted by one who possesses the original, the embedded information cannot be accessed by the user. For example, it would not be possible for a user's web browser to extract and display a caption or "property of" warning embedded in a downloaded image. The need to identify the original image before extraction also precludes oblivious, batch extraction. One might desire a web crawler or search engine to automatically find all illegal copies of any one of the many images belonging to, say, a particular photo archive, or all images with a certain embedded caption, but this is not possible with cover image escrow schemes (at least not without invoking computer vision). Finally, even assuming that the cover image has been identified and subtracted out, the proof value of such a watermark is questionable at best, since an "original" can always be constructed a posteriori to make any image appear to contain any watermark. The only practical application of cover image escrow schemes we have been able to identify is fingerprinting or traitor tracing[Pfi], in which many apparently identical copies of the cover image are distributed, but the owner wants to be able distinguish [illegible] identify users who have been giving away illegal copies.

The hiding methods presented in this paper are *oblivious*, meaning that the message can be read with no prior knowledge of the cover image. Other oblivious schemes have been proposed [BGM91, Cor95], but the information-theoretic limits on the problem have not been explicitly considered. We make comparisons between our hiding schemes and these other oblivious schemes later in the paper.

In the next section, we will estimate the amount of information that can be hidden (with minimal robustness) in an image as a function of signal-to-noise ratio. The bulk of the paper is a description of some new hiding schemes that fall short but are within a small constant factor of the theoretical hiding capacity. In the implementations of these schemes presented in this paper, we have chosen capacity over robustness, but we could have done otherwise. In the conclusion, we return to the discussion of modeling the trade offs between hiding capacity, perceptibility, and robustness using the quantities channel capacity, signal-to-noise, and process gain.

2 Channel Capacity

By Nyquist's theorem, the highest frequency that can be represented in our cover image is $\frac{1}{2}\frac{cycle}{pixel}$. The band of frequencies that may be represented in the image ranges from $-\frac{1}{2}\frac{cycle}{pixel}$ to $+\frac{1}{2}\frac{cycle}{pixel}$, and therefore the bandwidth W available for information hiding is $2 \times \frac{1}{2}\frac{cycle}{pixel} = 1\frac{cycle}{pixel}$.

For a channel subject to Gaussian noise, the channel capacity, which is an upper bound on the rate at which communication can reliably occur, is given by [SW49]

$$C = W \log_2(1 + \frac{S}{N})$$

Since the bandwidth W is given in units of pixel^{-1} and the base of the logarithm is 2, the channel capacity has units of bits per pixel. For some applications (particularly print) it might be desirable to specify the bandwidth in units of millimeters^{-1}, in which case the channel capacity would have units of bits per millimeter.

This formula can be rewritten to find a lower bound on the $\frac{S}{N}$ required to achieve a communication rate C given bandwidth W. Shannon proved that this lower bound is in principle tight, in the sense that there exist ideal systems capable of achieving communications rate C using only bandwidth W and signal-to-noise $\frac{S}{N}$. However, for practical systems, there is a tighter, empirically determined lower bound: given a desired communication rate C and an available bandwidth W, a message can be successfully received if the signal-to-noise ratio is at least some small *headroom factor* α above the Shannon lower bound. The headroom α is greater than 1 and typically around 3. [She95]

$$\frac{S}{N} \geq \alpha\left(2^{\frac{C}{W}} - 1\right)$$

In information hiding, $\frac{S}{N} < 1$, so $\log_2(1 + \frac{S}{N})$ may be approximated as $\frac{S/N}{\ln 2}$ or about $1.44\frac{S}{N}$. [She95] Thus $\frac{S}{N} \geq \frac{\alpha}{1.44}\frac{C}{W}$. So in the low signal-to-noise regime relevant to information hiding, channel capacity goes linearly with signal-to-noise.

The average noise power of our example cover image was measured to be 902 (in units of squared amplitude). For signal powers 1, 4, and 9 (amplitude^2), the channel capacity figures are 1.6×10^{-3} bits per pixel, 6.4×10^{-3} bits per pixel, and 1.4×10^{-2} bits per pixel. In an image of size 320 x 320, the upper bound on the number of bits that can be hidden and reliably recovered is then 320^2C. In our cover image of this size, then, using gain factors of 1, 2, and 3 (units of amplitude), the Shannon bound is 160 bits, 650 bits, and 1460 bits. With a headroom factor of $a = 3$, we might realistically expect to hide 50, 210 or 400 bits using these signal levels.

3 Modulation Schemes

In the modulation schemes we discuss in this paper, each bit b_i is represented by some basis function ϕ_i multiplied by either positive or negative one, depending on the value of the bit. The modulated message $S(x,y)$ is added pixel-wise to the cover image $N(x,y)$ to create the stego-image $D(x,y) = S(x,y) + N(x,y)$. The modulated signal is given by

$$S(x,y) = \sum_i b_i \phi_i(x,y)$$

Our basis functions will always be chosen to be orthogonal to each other, so that embedded bits do not equivocate:

$$< \phi_i, \phi_j > = \sum_{x,y} \phi_i(x,y)\phi_j(x,y) = nG^2\delta_{ij}$$

where n is the number of pixels and G^2 is the average power per pixel of the carrier.

In the ideal case, the basis functions are also uncorrelated with (orthogonal to) the cover image N. In reality, they are not completely orthogonal to N; if they were, we could hide our signal using arbitrarily little energy, and still recover it later.

$$< \phi_i, N > = \sum_{x,y} \phi_i(x,y)N(x,y) \approx 0$$

For information hiding, basis functions that are orthogonal to typical images are needed; image coding has the opposite requirement: the ideal is a small set of basis functions that approximately spans image space. These requirements come in to conflict when an image holding hidden information is compressed: the ideal compression scheme would not be able to represent the carriers (bases) used for hiding at all.

The basis functions used in the various schemes may be organized and compared according to properties such as total power, degree of spatial spreading (or localization), and degree of spatial frequency spreading (or localization). We will now explain and compare several new image information hiding schemes, by . . . . used

3.1 Spread Spectrum Techniques

In the spectrum-spreading techniques used in RF communications[Dix94, SOSL94], signal-to-noise is traded for bandwidth: the signal energy is spread over a wide frequency band at low SNR so that it is difficult to detect, intercept, or jam. Though the total signal power may be large, the signal to noise ratio in any band is small; this makes the signal whose spectrum has been spread difficult to detect in RF communications, and, in the context of information hiding, difficult for a human to perceive. It is the fact that the signal energy resides in all frequency bands that makes spread RF signals difficult to jam, and embedded information difficult to remove from a cover image. Compression and other degradation may remove signal energy from certain parts of the spectrum, but since the energy has been distributed everywhere, some of the signal should remain. Finally, if the key used to generate the carrier is kept secret, then in the context of either ordinary communications or data hiding, it is difficult for eavesdroppers to decode the message.

Three schemes are commonly used for spectrum spreading in RF communications: *direct sequence*, *frequency hopping*, and *chirp*. In the first, the signal is modulated by a function that alternates pseudo-randomly between $+G$ and $-G$, at multiples of a time constant called the chiprate. In our application, the chiprate is the pixel spacing. This pseudo-random carrier contains components of all frequencies, which is why it spreads the modulated signal's energy over a large frequency band. In frequency hopping spread spectrum, the transmitter rapidly hops from one frequency to another. The pseudo-random "key" in this case is the sequence of frequencies. As we will see, this technique can also be generalized to the spatial domain. In chirp spreading, the signal is modulated by a chirp, a function whose frequency changes with time. This technique could also be used in the spatial domain, though we have not yet implemented it.

3.2 Direct-Sequence Spread Spectrum

In these schemes, the modulation function consists of a constant, integral-valued gain factor G multiplied by a pseudo-random block ϕ_i of $+1$ and -1 values. Each block ϕ_i has a distinct location in the (x,y) plane. In both versions of direct sequence spread spectrum we have considered, the blocks ϕ_i are non-overlapping (and therefore trivially orthogonal); they tile the (x,y) plane without gaps. Because distinct basis functions ϕ_i do not overlap in the x and y coordinates, we do not need to worry about interference and can write the total power

$$P \equiv \sum_{x,y}^{X,Y}(\sum_i Gb_i\phi_i(x,y))^2 = \sum_i \sum_{x,y}^{X,Y}(Gb_i\phi_i(x,y))^2 = G^2XY = nG^2$$

The definition holds in general, but the first equation only holds if the ϕ_i tile the (x,y) plane without overlaps. Non-integral values of power can be implemented by "dithering": choosing step values

$$g \in (-G), (-G+1), \ldots, (-1), (0), (1), \ldots, (G-1), (G)$$

with probabilities $p(g)$ such that the average power $G^2 = \sum_g p(g)g^2$.

The embedded image is recovered by demodulating with the original modulating function. A TRUE (+1) bit appears as a positive correlation value; a FALSE (−1) bit is indicated by a negative correlation value. We have found the median of the maximum and minimum correlation values to be an effective decision threshold, though it may not be optimal. For this scheme to work, at least one value of the embedded image must be TRUE and one FALSE. In the version of direct sequence data hiding presented in [Cor95], a similar problem is avoided by including 0101 at the beginning of each line.

A more sophisticated scheme would be to use a "dual-rail" representation in which each ϕ_i is broken in two pieces and modulated with $(-1)(1)$ to represent FALSE and $(1)(-1)$ to represent TRUE. Then to recover the message, each bit can be demodulated twice, once with $(-1)(1)$ and once with $(1)(-1)$. Whichever correlation value is higher gives the bit's value. This dual rail scheme also has advantages for carrier recovery.

Bender et al.'s Patchwork algorithm[BGM91] for data hiding in images can be viewed as a form of spread spectrum in which the pseudo-random carrier is sparse (is mostly 0s) and with the constraint that its integrated amplitude be zero enforced by explicit construction, rather than enforced statistically as in ordinary spread spectrum schemes.

In the Patchwork algorithm, a sequence of random pairs of pixels is chosen. The brightness value of one member of the pair is increased, and the other decreased by the same amount, G in our terminology. This leaves the total amplitude of the image (and therefore the average amplitude) unchanged. To demodulate, they find the sum $S = \sum_{i=1}^{n} a_i - b_i$, where a_i is the first pixel of pair i, and b_i is the second pixel of pair i. Notice that because addition is commutative, the order in which the pixel pairs were chosen is irrelevant. Thus the set of pixels at which single changes are made can be viewed as the non-zero entries in a single two-dimensional carrier $\phi(x, y)$. Bender et al. always modulate this carrier with a coefficient $b = 1$, but $b = -1$ could also be used. In this case, the recovered value of s would be negative. If the same pixel is chosen twice in the original formulation of the Patchwork algorithm, the result is still a carrier $\phi(x, y)$ with definite power and bandwidth. Thus Patchwork can be viewed as a special form of spread spectrum (with extra constraints on the carrier), and evaluated quantitatively in our information-theoretic framework.

Fully Spread Version We have implemented a "fully spread" version of direct sequence spread spectrum by chosing a different pseudo-random ϕ_i for each value of i. This fully spreads the spectrum, as the second figure in the second column of Figure 2 shows. The figure shows both space and spatial frequency representations of the cover image, the modulated pseudo-random carrier, and the sum of the two, the stego-image.

To extract the embedded message (to demodulate), we must first recover the carrier phase. If the image has only been cropped and translated, this can be accomplished by a two dimensional search, which is simple but effective.

The point at which the cross-correlation of the stego-image and the carrier is maximized gives the relative carrier phase. We have implemented this brute force carrier phase recovery scheme, and found it to be effective. Rotation or scaling could also be overcome with more general searches.

Once the carrier has been recovered, we project the stego-image onto each basis vector ϕ_i:

$$o_i = < D, \phi_i > = \sum_{x,y} D(x, y)\phi_i(x, y)$$

and then threshold the o_i values. We have used the median of the maximum and minimum o_i value as the threshold value. Note that for this to work, there must be at least one $b_i = -1$ and one $b_i = +1$. Above we discussed more sophisticated schemes that avoid this problem. Figure 2 shows the original input to be embedded, the demodulated signal recovered from the stego-image, the threshold value, and the recovered original input.

Tiled Version This scheme is identical to the "fully spread" scheme, except that the same pseudo-random sequence is used for each ϕ_i. The ϕ_i differ from one another only in their location in the (x, y) plane. Unlike the fully spread version, which is effectively a one-time pad, some information about the embedded icon is recoverable from the modulated carrier alone, without *a priori* knowledge of the unmodulated carrier. This information appears as the inhomogeneities in the spatial frequency plane of the modulated carrier visible in Figure 3. If a different icon were hidden, the inhomogeneity would look different. One advantage of the tiled scheme is that carrier recovery requires less computation, since the scale of the search is just the size of one of the ϕ_i tiles, instead of the entire (x, y) plane. Given identical transmit power, this scheme seems to be slightly more robust than the "fully spread" scheme.

These two spread spectrum techniques are resistant to JPEGing, if the modulated carrier is given enough power (or more generally, as long as the jamming margin is made high enough). With carrier recovery, the two direct sequence schemes are resistant to translation and some cropping. However, unlike the frequency hopping scheme that we will describe below, the direct sequence basis functions are fairly localized in space, so it is possible to lose some bits to cropping.

Predistortion In addition to simply increasing the signal to improve compression immunity, Figure 4 illustrates a trick, called *predistortion*, for increasing the robustness of the embedded information when it is known that the image will be, for example, JPEG compressed. We generate the pseudo-random carrier, then JPEG compress the carrier by itself (before it has been modulated by the embedded information and added to the cover image), and uncompress it before modulating. The idea is to use the compression routine to filter out in advance all the power that would otherwise be lost later in the course of compression.[1] Then the gain can be increased if necessary to compensate for the power lost to compression. The once JPEGed carrier is invariant to further JPEGing using the same quality factor (except for small numerical artifacts).[2] Figure 4 shows both the space and spatial frequency representation of the JPEG compressed carrier. Note the suppression of high spatial frequencies. Using the same power levels, we achieved error-free decoding with this scheme, but had several errors using the usual fully spread scheme without the pre-distortion of the carrier. Tricks analogous to this are probably possible whenever the information hider has a model of the type of distortion that will be applied. Note that this version of predistortion cannot be applied to our next scheme, or to the version of direct sequence spread spectrum in [Cor95], because in these schemes carriers overlap in space and therefore interfere.

3.3 Frequency Hopping Spread Spectrum

This scheme produces perceptually nice results because it does not create hard edges in the space domain. However, its computational complexity, for both encoding and decoding, is higher than that of the direct sequence schemes.

Each bit is encoded in a particular spatial frequency; which bit of the embedded message is represented by which frequency is specified by the pseudo-random key. In our trial implementation of frequency hopping spread spectrum, however, we have skipped the pseudo random key, and instead chosen a fixed block of 10 by 10 spatial frequencies, one spatial frequency for each bit. One advantage of the frequency hopping scheme over the direct sequence techniques is that each bit is fully spread spatially: the bits are not spatially localized at all. This means that the scheme is robust to cropping and translation, which only induce phase shifts.

An apparent disadvantage of the frequency hopping scheme is that because the functions overlap in the space domain, the time to compute the modulated carrier appears to be kXY, where k is the number of bits, instead of just XY, the time required for the direct sequence schemes. However, the Fast Fourier Transform (more precisely, a Fast Discrete Cosine Transform) can be used to implement this scheme, reducing the time to $XY \log_2 XY$. This is a savings if $\log_2 XY < k$. In our example, $\log_2 320 \times 320 = 16.6$ and $k = 100$, so the FFT is indeed the faster implementation.

Figure 5 illustrates the frequency hopping modulation scheme. The results, shown in figure 6, are superior to the direct sequence schemes both perceptually and in terms of robustness to accidental removal. There is little need to threshold the output of the demodulator in this case. However, encoding and decoding require significantly more computation time.

This scheme survived gentle JPEGing[3] with no predistortion, as illustrated in figure 7.[4]

A disadvantage of this scheme for some purposes is that it would be relatively easy to intentionally remove the embedded message, by applying a spatial filter of the appropriate frequency. A more secure implementation of the scheme would disperse the frequencies from one another, to make this sort of filtering operation more difficult. The main disadvantage of this scheme relative to the direct sequence schemes is that, even using the FFT, its computational complexity for encoding and decoding is greater ($XY \log XY$ rather than XY).

4 Discussion

We have suggested that information and communication theory are useful tools both for analyzing information hiding, and for creating new information hiding schemes. We showed how to estimate the signal-to-noise needed to hide a certain number of bits given bandwidth W. A shortcoming of our channel capacity estimate is that we used the capacity formula for a Gaussian channel, which is not the best model of the "noise" in a single image, as a glance at any of the frequency domain plots in the figures will reveal. The Gaussian channel has the same power at each frequency, but clearly these images do not, especially after compression. A more refined theory would use a better statistical model of the image channel, and would therefore be able to make better estimates of the signal-to-noise needed to hide a certain number of bits. This would also lead to better hiding schemes, since the signal energy could be distributed more effectively.

The scheme we have called "frequency hopping" is superior perceptually, and in terms of robustness to accidental removal, to the direct sequence schemes with which we experimented. Direct sequence may be less vulnerable to intentional removal, and wins in terms of computational complexity.

Assuming that the Gaussian channel approximation discussed above is not too misleading, our capacity estimates suggest that there exist significantly better schemes than we have presented, capable of hiding several hundred bits in an image in which we hid one hundred. Hybrid modulation/coding schemes such as

[1] By compressing the carrier separately from the image, we are treating the JPEG algorithm as an operator that obeys a superposition principle, which it does in an approximate sense defined in the Appendix.

[2] It should be apparent from the description of JPEG compression in the Appendix that the output of the JPEG operator (or more precisely, the operator consisting of JPEG followed by inverse JPEG, which maps an image to an image) is an eigenfunction and in fact a fixed point of that operator, ignoring small numerical artifacts.

[3] All the JPEG compression reported here was done in Photoshop using the "high quality" setting.

[4] In fact, it is not possible to predistort in the frequency hopping scheme: because the basis functions overlap, the resulting interference pattern depends strongly on the particular values of the bits being encoded. There is no single pattern onto which we can project the stego-image to recover the embedded data; we must (naively) project it onto a sequence of vectors, or (more sophisticated) use the FFT. In either case the idea of predistortion does not apply, at least not in the same way it did in the non-overlapping direct sequence schemes.

trellis coding are a promising route toward higher hiding densities. But better models of channel noise (the noise due to cover images themselves, plus distortion) would lead immediately to better capacity estimates, and better hiding schemes.

In all the practical examples in this paper, we have tried to hide as much information as possible using a given signal-to-noise. However, keeping signal-to-noise and bandwidth fixed, communication rate can instead be traded for robustness to jamming. The quantities known as jamming margin and processing gain in spread spectrum communication theory are helpful in capturing this notion of robustness.

Processing gain is the ratio $\frac{W}{M}$ of available bandwidth W to the bandwidth M actually needed to represent the message. Jamming margin, the useful measure of robustness, is the product of signal-to-noise and processing gain. If the actual signal-to-noise ratio is $\frac{S}{N}$, then the jamming margin or effective signal-to-noise ratio $\frac{E}{J}$ after demodulation is given by $\frac{E}{J} = \frac{W}{M}\frac{S}{N}$. So robustness may be increased either by increasing signal-to-noise (at the cost of perceptibility, as we will explain in more detail below), or by decreasing the size of the embedded message (the capacity), which increases the processing gain. For example, in the case of our direct sequence schemes, the processing gain increases when we hide fewer bits because each bit can be represented by a larger block. The Patchwork hiding scheme referred to earlier sacrifices communication rate entirely (hiding just one bit) in order to buy as much robustness as possible.

Signal-to-noise ratio provides a rough estimate of perceptibility, because, all other things being equal, the higher the signal-to-noise, the more visible the modulated carrier will be. However, keeping signal-to-noise constant, some carriers—particularly those with mid-range spatial frequencies, our experience so far suggests—will be more more perceptible than others. So the crudest model of perceptibilty is simply signal-to-noise ratio; a plausible refinement might be the integral over all spatial frequencies of the signal-to-noise as a function of frequency weighted by a model of the frequency response of the human visual system. Methods for quantifying visibility to humans might be a new theoretical avenue to explore, and developing systematic methods for minimizing the visibility of hidden signals is certainly a challenge to information hiding practice. The pre-distortion technique demonstrated in this paper can be viewed as a first step in this direction, in the sense that successful compression schemes comprise implicit, algorithmic models of the human visual system (the ideal compression scheme would encompass a complete model of the human visual system). It will be interesting to watch the development of information hiding schemes and their co-evolutionary "arms race" with compression methods in the challenging environment of the human visual system.

A Approximate superposition property for JPEG operator

An operator O obeys superposition if $O(f+g) - (O(f) + O(g)) = 0$. Each coefficient generated by the JPEG operator J satisfies $-1 \leq J(f+g) - (J(f) + J(g)) \leq 1$. In other words, JPEGing a pair of images separately and then adding them yields a set of coefficients each of which differs by no more than one quantization level from the corresponding coefficient found by adding the images first and then JPEGing them (using the same compression parameters in both cases).

The proof is simple. For a gray scale image, the unquantized JPEG coefficients S_{ij} are found by expanding each 8×8 block in a cosine basis. The final quantized coefficients a_{ij} are found by dividing each S_{ij} by a quantization factor q_{ij} (where each q_{ij} is greater than one, since the purpose of the JPEG representation is to decrease the file size), and rounding toward zero[HB93]:

$$a_{ij} = \lfloor \frac{S_{ij}}{q_{ij}} \rfloor$$

The cosine expansion is a linear operation, and therefore obeys superposition, so (as long as $q_{ij} > 1$) we need only show that for any real numbers f and g, $-1 \leq \lfloor f+g \rfloor - \lfloor f \rfloor - \lfloor g \rfloor \leq 1$. Without loss of generality, we may take f and g to be non-negative and less than one, since the integer parts F and G of f and g satisfy $\lfloor F+G \rfloor - \lfloor F \rfloor - \lfloor G \rfloor = 0$. So, for such an f and g, $0 \leq f+g < 2$. There are now two cases to consider. If $0 \leq f+g < 1$, then $\lfloor f+g \rfloor - \lfloor f \rfloor - \lfloor g \rfloor = 0 - 0 - 0 = 0$. If $1 \leq f+g < 2$ then $\lfloor f+g \rfloor - \lfloor f \rfloor - \lfloor g \rfloor = 1 - 0 - 0 = 1$. Since $f+g < 2$, these are the only two cases. The case of f and g negative is analogous, yielding a discrepancy of either -1 or 0. The discrepancy in the case that f and g have opposite sign is less than in the same sign case. Therefore each a_{ij} coefficient produced by the JPEG operator satisfies our approximate superposition principle, $-1 \leq J(f+g) - (J(f) + J(g)) \leq 1$. Since each a_{ij} coefficient has a discrepancy of $+1$, 0, or -1, each S_{ij} has a discrepancy of $+q_{ij}$, 0, or $-q_{ij}$. Thus the total power of the deviation from superposition (in either the spatial frequency or pixel representation, by Parseval's theorem) is bounded above by $\sum_{ij} q_{ij}^2$. This explains why JPEGing the carrier separately from the cover image is a reasonable predistortion tactic.

Note that the more aggressive the compression (the larger the q_{ij} values), the larger the discrepancies, or deviations from superposition.

Acknowledgments

This research was performed in the laboratory of Neil Gershenfeld. The authors thank him for his advice and support. The second author thanks Joe Jacobson for his support. We thank Walter Bender, Dan Gruhl, and the News in the Future Consortium for introducing us to the problem of data hiding. We acknowledge the other members of the Physics and Media group, especially Joe Paradiso and Tom Zimmerman, for helpful conversations about modulation techniques. Maggie Orth made useful suggestions about the proof of the approximate superposition principle.

This work was supported in part by the MIT Media Lab's News in the Future Consortium, a Motorola Fellowship, the Hewlett-Packard Corporation, Festo Corporation, Microsoft, Compaq Computer Corporation, and the MIT Media Lab's Things That Think consortium.

References

[BGM91] W. Bender, D. Gruhl, and N. Morimoto. Techniques for data hiding. In *Proceedings of the SPIE*, pages 2420–2440, San Jose, CA, February 1991.

[BH93] M.F. Barnsley and L.P. Hurd. *Fractal Image Compression*. AK Peters, Ltd., Wellesley, Massachusetts, 1993.

[BOD95] F.M. Boland, J.J.K. O'Ruanaidh, and C Dautzenberg. Watermarking digital images for copyright protection. In *Proceedings, IEE International Conference on Image Processing and its Application*, Edinburgh, 1995.

[CKLS] I. Cox, J. Kilian, T. Leighton, and T. Shamoon. A secure, robust watermark for multimedia. *This volume*.

[Cor95] Digimarc Corporation. Identification/authentication coding method and apparatus. *U.S. Patent Application*, June 1995.

[Dix94] R.C. Dixon. *Spread Spectrum Systems with Commercial Applications*. John Wiley and Sons, New York, 1994.

[Pf] B. Pfitzmann. Trials of traced traitors. *This volume*.

[She95] T.J. Shepard. *Decentralized Channel Management in Scalable Multihop Spread-Spectrum Packet Radio Networks*. PhD thesis, Massachusetts Institute of Technology, July 1995.

[SOSL94] M.K. Simon, J.K. Omura, R.A. Scholtz, and B.K. Levitt. *The Spread Spectrum Communications Handbook*. McGraw-Hill, New York, 1994.

[SW49] C.E. Shannon and W.W. Weaver. *The Mathematical Theory of Communication*. The University of Illinois Press, Urbana, Illinois, 1949.

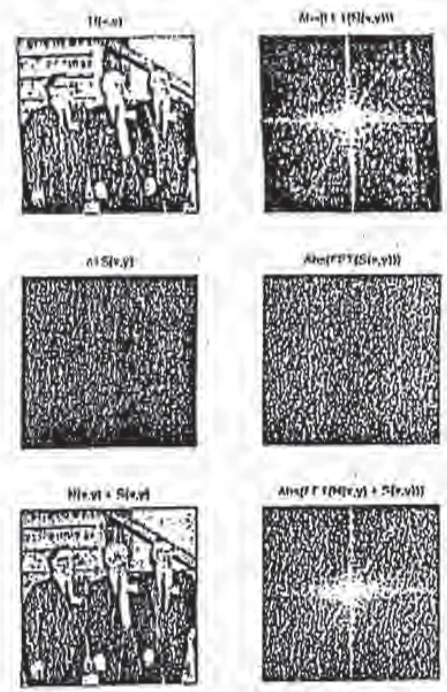


Fig. 1. "Fully Spread" version of direct sequence spread spectrum. The left column shows (from top to bottom) the space representation of the cover image, the modulated carrier, and the stego-image. The right column is the spatial frequency representation of the same three functions. The cover image has six bits of gray scale ($0-63$), and the power per pixel of this particular cover image, that is, the noise power per pixel, is $902 \approx 30^2$. The carrier alternates between $+2$ and -2 in this figure, so the signal power per pixel is $2^2 = 4$. We have added a constant c to the carrier to map the values into a positive gray scale.

Fig. 2. Demodulation of Fully Spread Scheme. Top: 100 bit input data icon to be embedded. Second: normalized values after demodulation. Third: threshold value. Bottom: Original input recovered by comparing demodulated values to threshold.

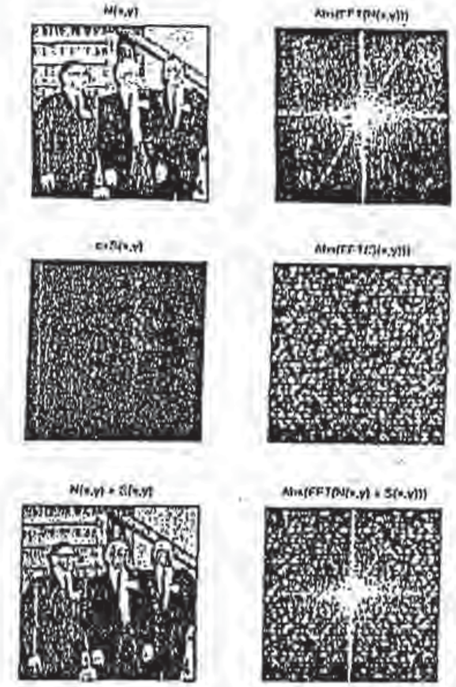


Fig. 3. Tiled version of spread spectrum modulation scheme. Note the inhomogeneities in the spatial frequency, view of the modulated carrier. As in the fully spread scheme, the noise power per pixel (the average power of the cover image) is 902, and the carrier ranges between +2 and −2, for a signal power of 4 per pixel.

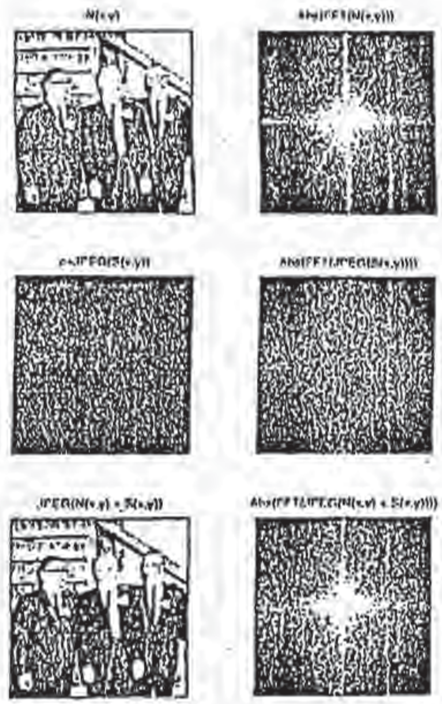


Fig. 4. Predistortion of carrier by JPEG compression to compensate for distortion from anticipated JPEG compression. The usual direct sequence carrier has been compressed and uncompressed before being used to modulate and demodulate. JPEG compression of the same quality factor will not alter the carrier further. The original average carrier power was 16; after JPEGing the carrier by itself, the average carrier power dropped to 8.8.

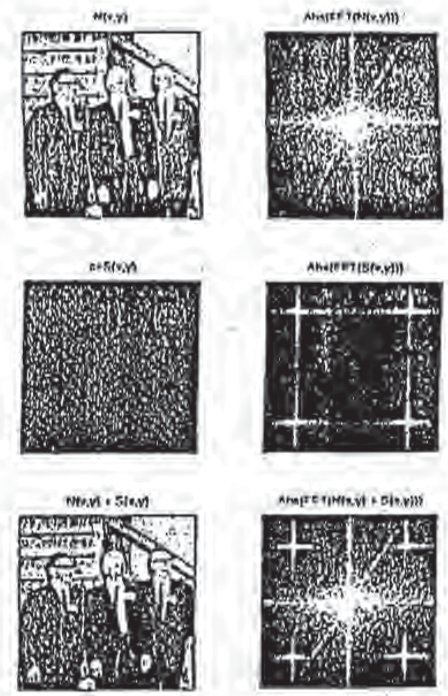


Fig. 5. Frequency Hopping spread spectrum. Average signal power = 8.1 (units of amplitude squared), and average noise power = 902.

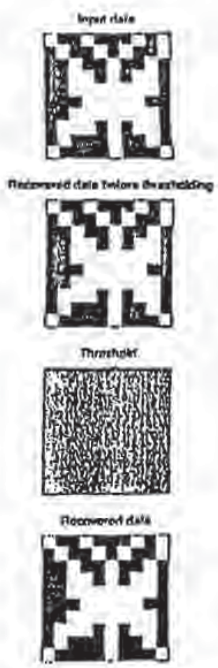


Fig. 6. Demodulation of Frequency Hopping spread spectrum.

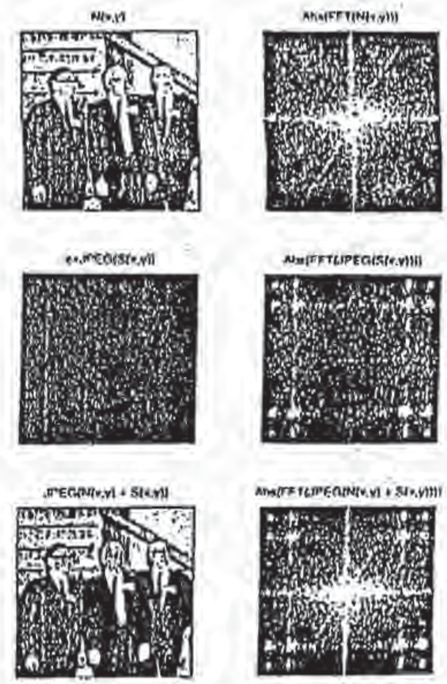


Fig. 7. Frequency Hopping spread spectrum, with JPEGed stego-image. The stego-image D was created, JPEGed at high quality, uncompressed, and then demodulated. To estimate the amount of signal lost in compression, we measured the average power of jpeg$(N + S) - N$ and found its value to be 5.6; the power in the carrier S was 9.1, as Figure 5 showed. The carrier shown for illustration purposes in the figure, labeled $e + JPEG(S(x, y))$, is in fact $JPEG(N + S) - N$. The carrier used to create the stego-image was in fact $S(x, y)$.

Fig. 8. Demodulation of Frequency Hopping spread spectrum, with JPEGed stego-image. The compression took its toll: contrast this output figure with the one from figure 6, which was so robust it needed no thresholding.

Watermarking Document Images with Bounding Box Expansion

Jack Brassil and Larry O'Gorman

{jtb|log}@bell-labs.com

Bell Laboratories
700 Mountain Ave.
Murray Hill, NJ 07972 USA

Imperceptible displacements of text objects has been shown to be a successful technique for hiding data in document images. In this paper we extend our earlier work to show how the height of a bounding box enclosing a group of text words can be used to increase the density of information hidden on a page. We present experimental results which show that bounding box expansions as small as 1/300 inch can be reliably detected, even after the distortions introduced by noisy image reproduction devices such as plain paper copiers. Digital watermarks based on this technique can be used with electronically disseminated documents for applications including copyright protection, authentication, and tagging.

1. Introduction

Traditional publishers seek access to the vast numbers of potential information consumers connected to computer networks such as the global Internet. However, many information providers remain reluctant to distribute their intellectual property electronically, in part due to their concerns about the unauthorized redistribution of their copyrighted materials. We have previously proposed a collection of techniques to discourage unauthorized copying of document images [1]. These techniques use digital watermarks created by imperceptible displacements of text objects in document images. Many other research groups are also successfully studying the use of digital watermarks in various media, including text, color image, audio and video [2, 3, 4, 5, 6, 7, 8, 9, 10]. In addition, a number of software companies have initiated efforts to pursue commercial applications of watermarks [11, 12, 13, 14, 15].

In this paper we introduce a new scheme to watermark binary document images containing text. Each document recipient receives either a paper or electronic document containing a set of *marks* constituting a unique fingerprint [16]. Each mark corresponds to the expansion of the height of a logical "bounding box" enclosing a group of adjacent characters (i.e. a text *block*) on a line. A bounding box is the smallest rectangle that encloses the block. We show how to encode documents imperceptibly with bounding box expansions, and demonstrate that this hidden information can be reliably recovered from degraded document images.

In the next section we briefly review our previous approaches to watermarking document images. Section 3 details our new approach to encoding and decoding documents with bounding box expansions. We also discuss troublesome image defects that characterize "noisy" document reproduction devices, as well as our approaches to circumventing these distortions. Section 4 presents experimental results that show that

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

610

(2)

Digital Signature of Color Images using Amplitude Modulation

Martin Kutter

Signal Processing Laboratory, EPFL
1015 Lausanne Switzerland

Frédéric Jordan

Signal Processing Laboratory, EPFL
1015 Lausanne Switzerland

Frank Bossen

Signal Processing Laboratory, EPFL
1015 Lausanne Switzerland

ABSTRACT

Watermarking techniques, also referred to as digital signature, sign images by introducing changes that are imperceptible to the human eye but easily recoverable by a computer program. Generally, the signature is a number which identifies the owner of the image. The locations in the image where the signature is embedded are determined by a secret key. Doing so prevents possible pirates from easily removing the signature. Furthermore, it should be possible to retrieve the signature from an altered image. Possible alternations of signed images include blurring, compression and geometrical transformations such as rotation and translation. These alterations are referred to as attacks. A new method based on amplitude modulation is presented. Single signature bits are multiply embedded by modifying pixel values in the blue channel. These modifications are either additive or subtractive, depending on the value of the bit, and proportional to the luminance. This new method has shown to be resistant to both classical attacks, such as filtering, and geometrical attacks. Moreover, the signature can be extracted without the original image.

Keywords: Watermarking, digital signature, copyright, color image, geometrical attack, steganography

1. INTRODUCTION

The emergence of digital imaging and of digital networks has made duplication of original artwork easier. In order to protect these creations, new methods for signing and copyrighting visual data are needed. Watermarking techniques, also referred to as digital signature, sign images by introducing changes that are imperceptible to the human eye but easily recoverable by a computer program. Generally, the signature is a number which identifies the owner of the image. The locations in the image where the signature is embedded are determined by a secret key. Doing so prevents possible pirates from easily removing the signature. Furthermore, it should be possible to retrieve the signature from an altered image. Possible alternations of signed images include blurring, compression and geometrical transformations such as rotation and translation. These alterations are referred to as attacks.

SPIE Vol. 3022 • 0277-786X/97/$10.00

Several watermarking algorithms have been developed in the past. Van Schyndel et al.[7] and Bender et al.[1] proposed a straightforward technique to sign gray scale images by adding a watermark image to the original image. A modification of the dithering rule was suggested by Matsui and Tanaka.[4] Another approach is based on the modification of DCT coefficients within a JPEG or an MPEG encoder.[2]

The main drawback of these early techniques is the lack of robustness to attacks. More recently, a spread spectrum technique has led to significant improvements.[3] Although it resists to filtering, it is vulnerable to geometrical attacks such as rotation, translation and image composition.

A new method based on amplitude modulation is presented. Single signature bits are multiply embedded by modifying pixel values in the blue channel. These modifications are either additive or subtractive, depending on the value of the bit, and proportional to the luminance. This new method has shown to be resistant to both classical and geometrical attacks. Moreover, the signature can be extracted without the original image.

This paper is structured as follows. Section 2 gives an overview of the new method. First the single embedding and retrieval of a single bit is described. This process is then generalized to multiple embedding of the same bit and to embedding of multiple bits. Section 3 describes how robustness to geometrical attacks is achieved. Section 4 shows some results and finally some conclusions are drawn in section 5.

2. ALGORITHM OVERVIEW

The main requirements for a digital signature are both invisibility to the human eye and robustness to alterations. To comply with the first requirement the signature is embedded in the blue channel, which is the one the human eye is least sensitive to. Also, changes in regions of high frequencies and high luminance are less perceptible, and thus favored. Robustness is achieved by embedding the signature several times at many different locations in the image.

First the single embedding and retrieval of a single bit is described. This process is then generalized to multiple embedding of the same bit and to embedding of multiple bits.

2.1. Single bit embedding

Let s be a single bit to be embedded in an image $I = \{R, G, B\}$, and $p = (i, j)$ a pseudo-random position within I. This position depends on a secret key K, which is used as a seed to the pseudo-random number generator. The bit s is embedded by modifying the blue channel B at position p by a fraction q of the luminance $L = 0.299R + 0.587G + 0.114B$ as:

$$B_{ij} \leftarrow B_{ij} + (2s - 1)L_{ij}q \tag{1}$$

where q is a constant determining the signature strength. The value q is selected such as to offer best trade-off between robustness and invisibility.

2.2. Single bit retrieval

In order to recover the embedded bit, a prediction of the original value of the pixel containing the information is needed. This prediction is based on a linear combination of pixel values in a neighborhood around p. Empirical results have shown that taking a cross-shaped neighborhood gives best performance. The prediction $\hat{B}_{ij}$ is thus computed as:

$$\hat{B}_{ij} = \frac{1}{4c}\left(\sum_{k=-c}^{c} B_{i+k,j} + \sum_{k=-c}^{c} B_{i,j+k} - 2B_{ij}\right) \tag{2}$$

where c is the size of the cross-shaped neighborhood.

To retrieve the embedded bit the difference δ between the prediction and the actual value of the pixel is taken:

$$\delta = B_{ij} - \hat{B}_{ij} \tag{3}$$

The sign of the difference δ determines the value of the embedded bit.

The embedding and the retrieval functions are not symmetric, that is the retrieval function is not the inverse of the embedding function. Although correct retrieval is very likely, it is not guaranteed. To further reduce the probability of incorrect retrieval, the bit is embedded several times, as described in the next section.

2.3. Multiple embedding

To improve retrieval performance, the bit can be embedded n times at different locations. These n positions $p_1, \ldots, p_n$ are determined by a pseudo-random sequence. As before the pseudo-random number generator is initialized with a seed equal to the secret key K.

By using a density parameter ρ, the redundancy control can be made image size independent. This density gives the probability of any single pixel being used for embedding. This value thus lies between 0 and 1, where 0 means that no information is embedded, and 1 that information is embedded in every pixel. The number of pixels used for embedding is equal to ρ times the total number of pixels in the image.

The locations for embedding are determined as follows: for each pixel of the image, a pseudo-random number x is generated. If x is smaller than ρ, then information is embedded into the pixel. Otherwise the pixel is left intact. In this process the scanning order is modified to make it image size independent. Instead of scanning the image line by line, column by column, a zig-zag like path is taken, as illustrated in figure 1.

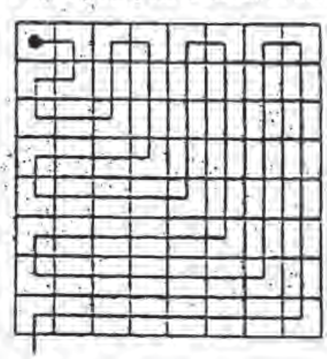


Figure 1: Modified image scanning order

To retrieve the bit, the difference between the prediction and the actual value of the pixel is computed for each p_k:

$$\delta_k = B_{p_k} - \hat{B}_{p_k} \tag{4}$$

These differences are then averaged:

$$\bar{\delta} = \frac{1}{\rho|I|}\sum_i \delta_i \qquad (5)$$

where $|I|$ is the number of pixels contained in image I.

The sign of the average difference $\bar{\delta}$ determines the value of the multiple embedded bit.

2.4. Extension to an m-bit signature

The extension to an m-bit signature $S = \{s_0, \ldots, s_{m-1}\}$ is straightforward: let $p_1 \ldots p_n$ be the n positions selected for the multiple embedding of a single bit. For each of these positions a signature bit is randomly selected and embedded.

Given an $m-2$ bit string to be embedded, 2 bits are added to the string to form an m-bit signature. These two bits are always set to 0 and 1, respectively. There are two reasons to do so:

1. it allows to define a threshold τ which improves signature retrieval
2. it defines a geometrical reference which is used to counter geometrical attacks, such as rotation, cropping, translation

These items are further described in the next sections.

2.5. Adaptive threshold

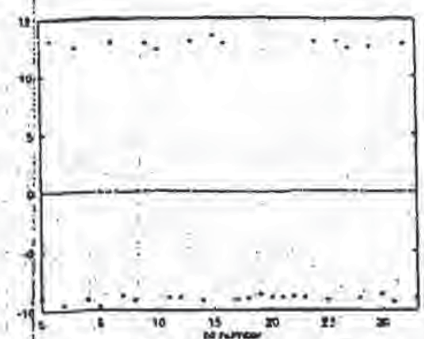
Considering each difference $\bar{\delta}^k$ that is used for information retrieval, the left graph in figure 2 clearly shows that the sign of $\bar{\delta}^k$ is a very good decision function. However, the right graph suggests that after an attack, this is not so anymore. Therefore the decision function needs to be adapted. Since it is known that the two first bits of the signature have values 0 and 1, respectively, this information can be used to compute an adaptive decision threshold. This threshold is defined as the average between $\bar{\delta}^0$ and $\bar{\delta}^1$:

$$s^k = \begin{cases} 1 & \text{if } \bar{\delta}^k > \frac{\bar{\delta}^0+\bar{\delta}^1}{2} \\ 0 & \text{otherwise} \end{cases} \qquad (6)$$

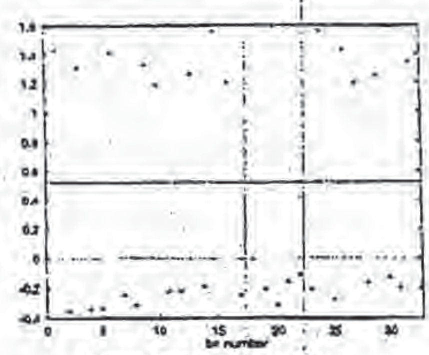
3. ROBUSTNESS TO GEOMETRICAL ATTACKS

In order to resist to geometrical attacks, it is required for the recovering algorithm to be able to determine what operation (translation, rotation) has been applied to produce the tampered image. To estimate this transform, a reference is needed. The two first bits of the signature can fulfill this requirement. Since these bits always have the same value, a known pattern is hidden within the image. By looking for this pattern the transform can be found.

Let G be the transform applied to the signed image to obtain the tampered image J: $J = G(I)$. For now, it is assumed that the transform G is affine. Let (i,j) be the position of a pixel in J. The corresponding pixel in J



Signed image



Signed image after attack

Figure 2: Behavior of $\bar{\delta}^s$ (delta) before and after an image attack (low pass filtering)

is at position $(\bar{i}, \bar{j})$ and is related to (i, j) by:

$$\begin{pmatrix} \bar{i} \\ \bar{j} \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & d \\ c & d & e \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} i \\ j \\ 1 \end{pmatrix} \tag{7}$$

where $a, \ldots, e$ are the transform parameters.

In order to retrieve the signature, the inverse G^{-1} of G is needed. By applying G^{-1} to J the image I is recovered and the signature can then be extracted.

The transform G^{-1} can be found by looking for the pattern created by the two first bits of the signature. Let H be an estimation of G^{-1}, and I_H the image obtained by applying H to J. Let's first suppose that H is equal to G^{-1}. The two first bits of the signature can clearly be retrieved from I_H. Also, the confidence of the retrieval is very high, that is the difference between $\bar{\delta}^1$ and $\bar{\delta}^0$ is maximum. Suppose now that H is slightly different from G^{-1}. The signature can still be retrieved but the difference between $\bar{\delta}^1$ and $\bar{\delta}^0$ is smaller than before. This difference gets smaller as the divergence between H and G^{-1} grows.

The difference can thus be used as an optimization criterion $\varphi(H)$ defined as $\varphi(H) = \bar{\delta}^1(I_H) - \bar{\delta}^0(I_H)$. As mentioned before $\varphi(H)$ is maximal for $H = G^{-1}$. However the function $\varphi(H)$ is not a smooth function. Optimization methods such as gradient descent would thus not be suitable. In this case full search methods have to be used.

The search can be sped up a lot if the nature of the transform G is given. For instance if it is known that the transform is a pure translation or rotation, the search space is greatly reduced.

4. RESULTS

To confirm the invisibility of the embedding process, an image (640 by 480 pixels, 24-bit color) with blue tones has been signed (see figure 3). For this particular image, the parameters have been set as follows: the signature S is 34 bit long and its value is the 32-bit number 1234567890 augmented by the two constant bits 0 and 1, the embedding density ρ is 0.55, the embedding strength is given by $q = 0.1$, and the size of the crossed-shape window is $c = 3$.

Original image

Signed image

Figure 3: Invisible embedding of information

To verify the robustness of the proposed method, the signed image has been attacked in several ways, namely:

- blurring
- JPEG encoding/decoding
- rotation
- composition with another image

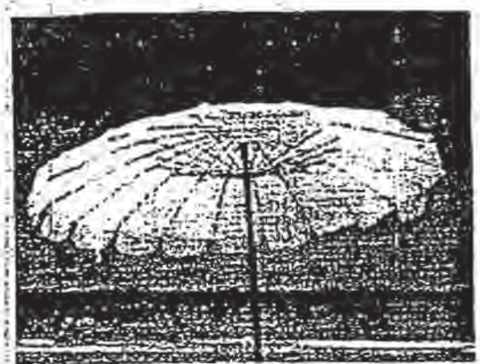
The next sections describe more precisely each of these attacks. Although in this document comprehensive results are only provided for these four attacks, the method has also shown to be resistant to other attacks including pixel spreading, pixelizing, color quantization, translation, cropping, and despeckling (median filtering).

4.1. Blurring

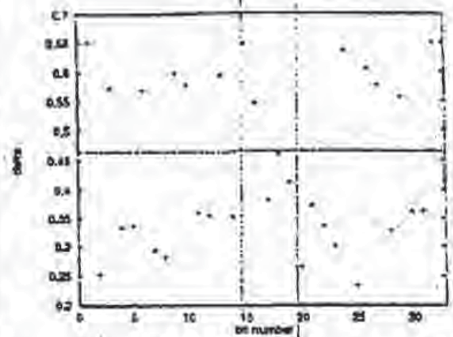
Figure 4 shows the signed image after blurring. The blurring function is as follows. Each color value is replaced by the average value of pixels within a 5 by 5 neighborhood.

The graph on the right hand side of figure 4 clearly indicates that the strength of the signature is much lowered by the attack. The average absolute difference between the δ^{A} and the threshold τ goes down from about 10 before the attack (see figure 2) to about 0.1 after the attack.

Although the signature is correctly retrieved after the blurring, the limits of the proposed method appear. Indeed the δ^{16} lies very close to the threshold and blurring the image even more would probably result in an erroneous signature retrieval. However the image quality would then also be much lower.



Blurred image



Retrieval quality

Figure 4: Simple image attack: blurring

4.2 JPEG encoding

Figure 5 shows the signed image after a JPEG encoding/decoding cycle. The quality factor for JPEG compression was set to 75 percent, which is the default value. Again the average absolute difference between the δ^i and the threshold r is much lower than before the attack. However each δ^i clearly lies on one of the sides of the threshold, and the signature can thus be correctly retrieved with great confidence.

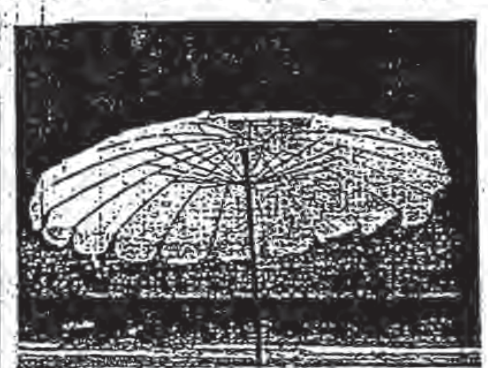
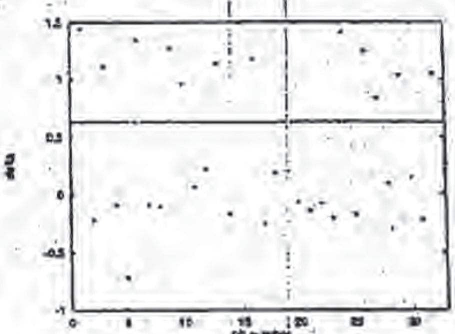


Image after JPEG compression/decompression

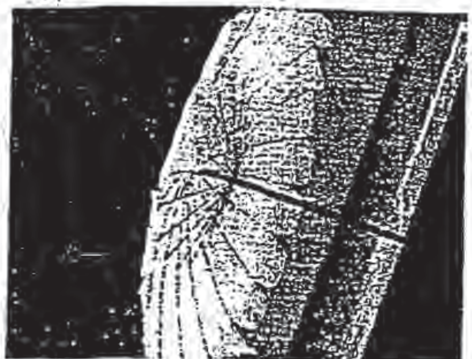


Retrieval quality

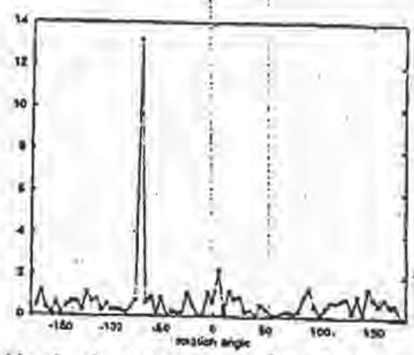
Figure 5: JPEG attack

4.3. Rotation

To test the robustness of the proposed method against geometrical attacks, the signed image has been rotated to the left by an angle of 70 degrees (see figure 6). Considering that it is known that the attack is purely rotational, $q(H)$ is computed for every H defined as a rotation between 0 to 360 degrees, with increments of 5 degrees. The graph in figure 6 shows how the value $q(H)$ is affected by the angle. Clearly the optimum lies at -70 degrees, that is, the amount by which the signed image was rotated. The signature can thus accurately be retrieved. Although a full search technique is used, the retrieval is still quite fast. Less than 20 seconds[1] were needed to do so.



The signed image rotated 70 degrees to the left



Searching for the transform: $q(H)$ as a function of the rotation angle

Figure 6: Geometrical attack: rotation

4.4. Composition with another image

Figure 7 shows an example of image composition. Given two signed images, each being signed with a different key, a third one is created by taking some pixels from the first one and some from the second one. This procedure can also be seen as a mixture of cropping and translation.

In this case, the algorithm is able to correctly retrieve both signatures given the appropriate keys.

5. CONCLUSION

A novel technique for image watermarking has been presented. The signing process has shown to be unnoticeable to the human eye. It has also been demonstrated that the signature is immune to a variety of attacks, including filtering, compression, and geometrical transforms. The resistance to the latter kind of attack without the need for the original image is the main improvement brought by this new method.

The proposed algorithm could be improved in several ways. First, all color channels could be exploited. The strength of the signature in each channel would be proportional to the sensitivity of the human eye to it. Also, robustness could be improved with the use of optimal error correcting codes. The current algorithm already features a primitive error correcting code based on the multiplicity of the embedding. However, it is well known that redundancy codes are far from optimality.

The authors would like to thank Vincent Vaerman for providing the images.

[1] On a Sun Ultra 1 workstation

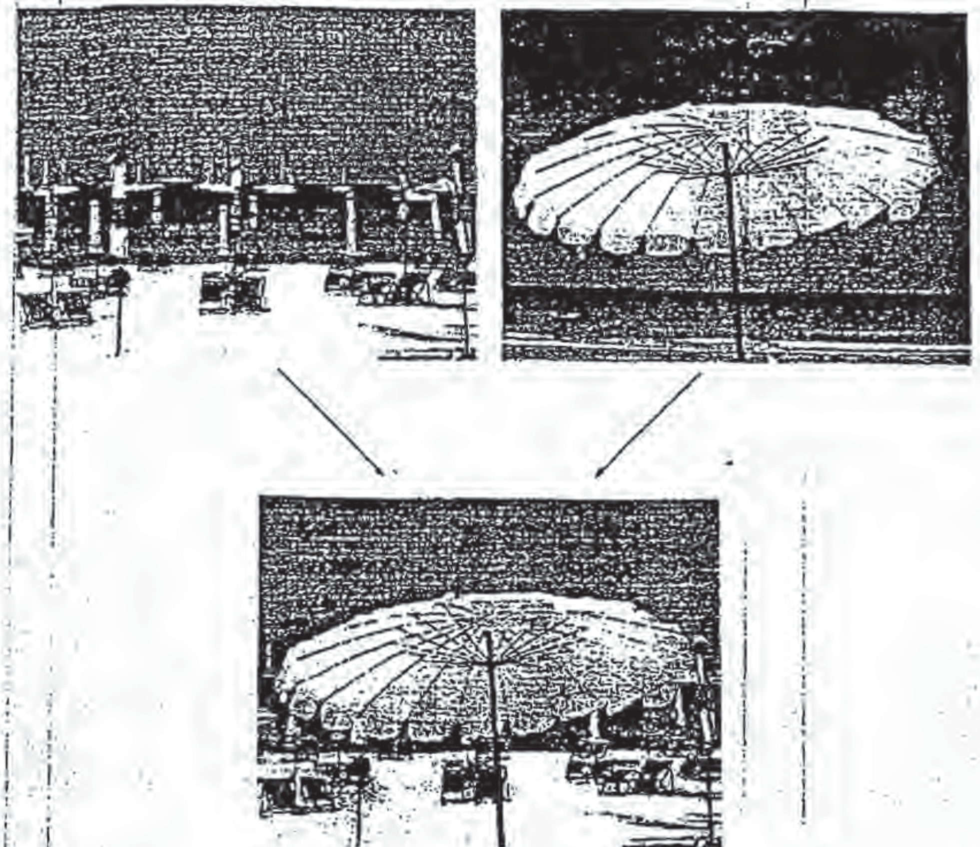


Figure 7: Composition example

6. REFERENCES

[1] W. Bender, D. Gruhl, and N. Mormoto. Techniques for data hiding. In *SPIE*, volume 2420, February 1995.

[2] S. Burgett, E. Koch, and J. Zhao. A novel method for copyright labelling digitized image data. *IEEE Transactions on Communications*, September 1994.

[3] I. Cox, J. Kilian, T. Leighton, and T. Shamoon. Secure spread spectrum watermarking for multimedia. Technical Report 95-10, NEC Research Institute, 1995.

[4] K. Matsui and K. Tanaka. Video-steganography: How to secretly embed a signature in a picture. *Journal of the interactive Multimedia Association Intellectual Property Project*, 1(1):187–206, January 1994.

[5] R.G. van Schyndel, A.Z. Torkel, and C.F. Osborne. A digital watermark. In *1st IEEE International Conference on Image Processing*, volume 2, pages 86–90, 1994.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☑ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

Using fractal compression scheme to embed a digital signature into an image

Joan Puate, Fred Jordan

Swiss federal institute of technology
Signal Processing Laboratory
CH-1015 Lausanne
Switzerland
Email: jordan@ltssg3.epfl.ch
Tel.: +41 21 693 70 89
FAX: +41 21 693 70 90

ABSTRACT

With the increase in the number of digital networks and recording devices, digital images appear to be a material, especially still images, whose ownership is widely threatened due to the availability of simple, rapid and perfect duplication and distribution means. It is in this context that several European projects are devoted to finding a technical solution which, as it applies to still images, introduces a code or Watermark into the image data itself. This Watermark should not only allow one to determine the owner of the image, but also respect its quality and be difficult to remove. An additional requirement is that the code should be retrievable by the only mean of the protected information. In this paper, we propose a new scheme based on fractal coding and decoding. In general terms, a fractal coder exploits the spatial redundancy within the image by establishing a relationship between its different parts. We describe a way to use this relationship as a means of embedding a Watermark. Tests have been performed in order to measure the robustness of the technique against JPEG conversion and low pass filtering. In both cases, very promising results have been obtained.

Keywords: digital signature, watermarking, image, copyright protection, security, fractal compression, IFS (Iterated Function Systems), FVT (Fractal Vector Technique), compression, internet

I. Introduction

1.1. The context

The last decades have seen exceptional development in the field of multimedia systems. Hence, people's needs will probably become very dependent on this phenomenon, and in order that the true potentials of these systems be properly exploited, some security mechanisms for privacy and intellectual rights must be given.

In the case of the protection of still images, finding a general solution is particularly difficult. The increasing number of digital networks and recording devices makes it very easy to create, distribute and copy such material. For these reasons, the demand for technical solutions against piracy is rapidly increasing among creators of multimedia information.

I.2. Several approaches

Different approaches have been already presented, they all intend to face mainly format conversions, low pass filtering and data compression. Whereas some works are based on the direct modification of the pixels' luminance [1,2,3,7], some others make use of a predicted coding scheme [5] or a JPEG compression scheme [6]. In [4], the mark is embedded in the LSB (Least Significant Bit) of the pixels' values and in [8], the use of a frequency modulation is done. Also, some techniques have been developed for video data, which is the case of [9].

I.3. Our proposal

In the following we propose a novel approach to Watermarking, based on the fractal theory of iterated transformations. For an image to sign we will construct its 'fractal code' in such a way that the decoded one includes a signature. Therefore, the signing algorithm consists of a coding-decoding process and retrieving the signature will be performed as a fractal coder. A signature thus obtained will have the important characteristic of being undetectable without the appropriate key.

Through this paper, some general concepts related to fractal coding techniques will be first explained, followed by their practical applications to our coder and decoder. Afterwards, we will introduce the signing and retrieving techniques as well as some results, focusing mainly on their robustness against JPEG compression and Blurring (3x3 kernel) attack. Finally, new ideas and possible improvements to be performed will be discussed.

II. Fractal Image Coding

II.1. Some general concepts

The fractals theory has proved to be suitable in many fields and particularly interesting in various applications of image compression. First important advances are due to M. F. Barnsley who introduces for the first time the term of Iterated Function Systems (IFS) [10,11,12,13] based on the self-similarity of fractal sets. Barnsley's work assumes that many objects can be closely approximated by self-similarity objects that might be generated by use of IFS simple transformations. From this assumption, the IFS can be seen as a relationship between the whole image and its parts, thus exploiting the similarities that exist between an image and its smaller parts.

At that point, the main problem is how to find these transformations or, what is the same, how to define the IFS. There is, in fact, a version of the IFS theory, the Local Iterated Function Systems theory, that minimizes the problem by stating that the image parts do not need to resemble the whole image but it is sufficient for them to be similar to some other bigger parts in it.

It was Arnaud E. Jacquin who developed an algorithm to automate the way to find a set of transformations giving a good quality to the decoded images [14]. In Fractal coding methods based on Jacquin's work, the main idea is to take advantage of the fact that different parts of the image at different scales are similar. As a matter of fact, they are block-based algorithms that intend to approximate blocks of a determined size with contractive transformations applied on bigger blocks. However, in theory the shape of the segments to encode is not restricted.

II.2. The basic theory

The main idea of a fractal based image coder is to determine a set of contractive transformations to approximate each block of the image (or a segment, in a more general sense), with a larger block.

Some basic aspects of the theory are given in the lines below (a clear and brief explanation can be found in [15] and [16]):

Let's consider a metric space ($\mathcal{E}$, d) where d is a given metric and $\mathcal{E}$ might be the space of the digital images. We can talk of a contractive transformation $\beta : \mathcal{E} \rightarrow \mathcal{E}$, when:

$$d(\beta(x), \beta(y)) \leq s\,d(x, y), \quad x, y \in \mathcal{E}, \quad 0 \leq s < 1$$

In this case, exists a point x^* such that:

$$B(x^*) = x^*$$
$$\lim_{n\to\infty} B^n(x) = x^* \ , \ \forall x \in \mathcal{E}$$

This point is called a fixed point.
An IFS consists of a complete metric space ($\mathcal{E}$, d) and a number of contractive mappings B_i defined on $\mathcal{E}$. The fractal transformation associated with an IFS is defined as:

$$B(E) = \bigcup_{i=1}^{N} B_i(E)$$

where E is any element of the space of non-empty compact subsets of $\mathcal{E}$.
If B_i is contractive for every i, then B is contractive and there is a fixed point for which:

$$A = B(A) = \bigcup_{i=1}^{N} B_i(A)$$

and

$$\lim_{n\to\infty} B^n(E) = A$$

A is called the *attractor* of IFS and the transformations are usually chosen to be affine.

Once B is determined, it is easy to get the decoded image by making use of the Contraction Mapping Theorem : the transformation B is applied iteratively on any initial image until the succession of images does not vary significantly.
However, given a set M, how to find a contractive transformation B such that its attractor A is close to M?
To answer this question we have to apply to the *Collage Theorem:*
For a set M and a contraction B with attractor **A**:

$$h(M, A) \le \frac{h(M, B(M))}{1-s}$$

Where h is the *Hausdorff Distance.*
That is to say that we can guarantee that M and A will be sufficiently close if we can make M and B(M) close enough.
In terms of B_i, and combining the two following expressions:

$$B(M) \approx M \ ; \qquad B(M) = \bigcup_{i=1}^{N} B_i(M)$$

we get

$$\bigcup_{i=1}^{N} B_i(M) = M$$

So, if we make a partition of M:

$$M = \bigcup_{i=1}^{n} m_i$$

then, m_i can be closely approximated by applying a contractive affine transformation B_i on the whole M:

$$m_i = B_i\ (M)$$

The theory of IFS was extended to Local IFS where each part of the image is approximated by applying a contractive affine transformation on another part of the image:

$$m_i = B_i\ (D_i)$$

where D_i is the bigger part from which m_i is approximated.

III. Algorithm description

The main idea to automate the searching of a Local IFS relies on a partition of the image in blocks of a fixed size, called *Range Blocks*. These blocks are then approximated from larger blocks, called *Domain Blocks*. The transformations normally applied on the Domain Blocks are contracting and luminance scaling and shifting. Some other isometric transformations are sometimes used.

We have used an algorithm based on Jacquin's work as the first step of the signing technique. In the following, a brief explanation of it is given.

III.1. Coder

Let O denote the image we want to encode. Let also O_r denote a partition of O in *nxn* blocks referred to as *Range Blocks (Rb)*. Similarly, O_d will denote another partition of O, this time in *2nx2n* blocks or *Domain Blocks (Db)* in steps of *nxn* pixels. The goal of the encoding algorithm is to establish a relationship between O_r and O_d in such a way that any Rb can be expressed as a set of transformations to be applied on a particular Db. The transformations that have been considered are *Contraction*, *Isometric transformation* (one out of eight), *Luminance Scaling* and *Luminance Shifting*. For each Rb in O_r denoted as Rb_j, the code will consist of a vector V_j and the appropriate transformations T_j, in such a way that:

- V_j has its origin in Rb_j and points to the correspondent Db_j which now becomes its *Matching block* (Mb_j).
- T_j if applied on Mb_j, minimizes the Mean Square Error (MSE) with respect to Rb_j.
- The couple (V_j, T_j) is the best solution (in the sense of the MSE) within a local area surrounding Rb_j in which we search for Mb_j.

The region of O_d where the search of Mb_j is performed is commonly taken as a square region surrounding the Rb_j. We will name this region LSR (Local Searching Region). The use of such a shape in the Matching Block determination might be justified from spatial redundancies considerations and that is essentially true. But that does not mean that other shapes can not give more than acceptable results on the Ranges Blocks approximation. Next figure shows the square surrounding region and a possible alternative:

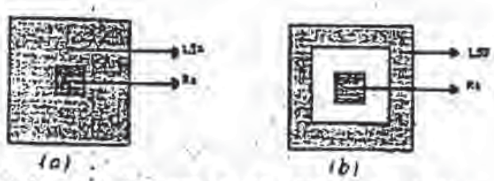


Fig. 1: (a) A square LSR and (b) an alternative solution

As we will describe further down, the assumption of this property will allow to make of this point the basis of our Watermarking proposal.

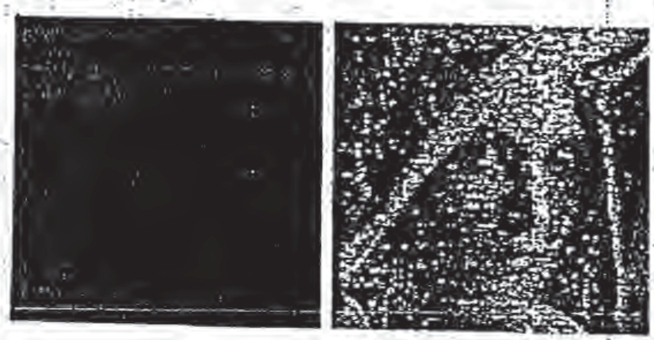
III.2. Decoder

Let's consider an initial image S with the only constraint that it has to be of the same size as O. As before, we consider a nxn-partition S_r in Rb, and a 2nx2n partition S_d in Db. The decoding algorithm takes for each Rb_j its Matching Block (pointed by V_j), applies on it the transformations (defined by T_j) and places the result back on Rb_j. These operations are performed for every Rb_j of S, and through some iterations (typically four). After each iteration a new image Q_i is obtained and it turns out that Q_i converges to O, according to the *Collage Theorem*. An important point is that the solution $\{V_j, T_j\}$ obtained for O remains exactly the same for Q_i. That is to say that for every Range Block the same Matching Block as before is found. We will also take advantage of this point to embed the signature by a properly choice of the vectors V_j.

Figure 2 shows some iterations for image Lena, when S_0 has been taken as a black image, n being equal to 4.

Figure 3, on the other hand, shows some iterations for image Lena, S_0 being a black image and n equal to 8.

It can be observed a better quality when n=4, above all in those parts of great detail. However, for n=4 the compression rate is much lower than for the case n=8. Therefore, there is choice to be made between quality and rate of compression. An intermediate solution might combine 4x4-Rangeblocks with 8x8-Rangeblocks. A quadtree based algorithm might achieve this compromise.



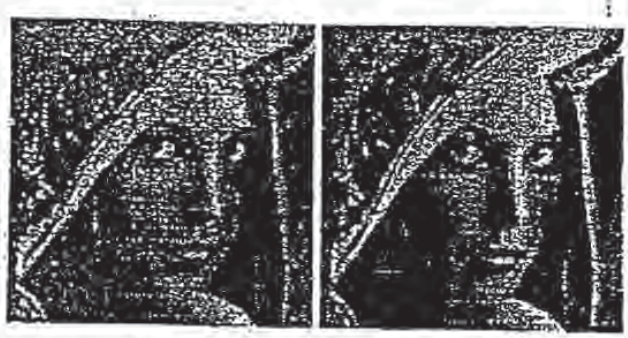


Fig. 2. Iterations 1,2,4 of a code for "Lena" applied on a gray image (n=4).

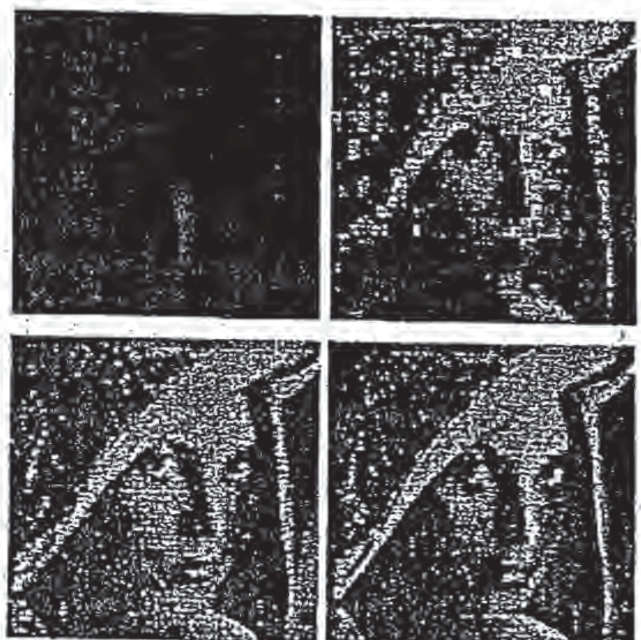


Fig. 3. Iterations 1,2,4 of a code for "Lena" applied on a gray image (n=8).

III.3. Signer

Signing an image consists of a coding-decoding process with variable searching regions:
Let's consider two different LSR, A and B (Fig. 4), and a third one, C, defined as their union (a different choice of the regions could have been made, perhaps as a function of the characteristics of the image). Let also $S=\{s_0, \ldots, s_{31}\}$ be a 32-bit signature. We will embed every bit with a redundancy U. The coding process is as follows:

- For each bit s_i U Range Blocks are randomly chosen and denoted by $\{Rb\}_i$. The random function used to get the blocks makes use of a 'seed' that should only be known by the user.
- If $s_i = 1$, $\{Rb\}_i$ is coded by searching for $\{Mb\}_i$ in regions $\{A\}_i$.
- If $s_i = 0$, $\{Rb\}_i$ is coded by searching for $\{Mb\}_i$ in regions $\{B\}_i$.
- The rest of Rb_j are coded by searching for Mb_j in $\{C\}_i$. Note that this would be the case for all Range Blocks in a non-signed image.

Then the decoding is performed as described above. The resulting image contains the signature.

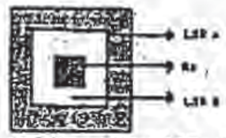


Fig. 4: A range Block, its LSR_A and its LSR_B. LSR_C is defined as their union.

III.4. Retriever

The whole fractal code of an image can be expressed as the union of every Range Block single code:

$$m = \bigcup_j (V_j, T_j)$$

Likewise, for an image Q obtained after an iterative application of μ on any initial image, we consider its fractal code π. Since, in Q, every Range block is not just an approximation to a transformed Domain Block but it is exactly a transformed Domain Block, it turns out that $\pi = \mu$.

Thus, we will be able to identify the signature by simply accessing the Range Blocks of Q defined by the 'seed' used when signing, recoding them and checking the values of V_j.

The rule to decide if a Range Block has been signed with a zero or a one, is the following one:

- If V_j belongs to region A_j, then a one has been embedded.
- If V_j belongs to region B_j, then a zero has been embedded.

In normal conditions, there ought to be a number of U recognition of bit one for those bits one in the signature, and of U recognition of bit zero for those bits zero in the signature.

To make the final decision there is a need of a threshold. It is going to be defined as the mean of the results obtained for bits two and three of the signature. Thus, they are always being embedded as a one and a zero, respectively.

IV. Results

This section is divided in two parts. First one concerns the case for n=4, and second one the case for n=8. In both, the robustness to JPEG compression and to low pass filtering (3x3-kernel blurring) is discussed, as well as the quality of the signed images against the original and the non signed but fractal encoded.

All tests have been performed by embedding a 32-bits signature in 'Lena' image (256x256), then applying the retriever. The chosen signature has a value of one in even bits and zero in odd bits. The redundancy U is equal to 50 for the case n=4, and equal to 25 for the case n=8.

The LSRs have been defined as follows:

LSR_A $k=\pm6$; $l=\pm6$;

LSR_B $|k|\leq5$; $|l|\leq5$;

Where (k,l) are the coordinates of vector V.

Moreover, we have defined a parameter P as the ratio between the number of vectors found in region A and the redundancy U. Since each bit equal to one has been embedded in LSR A, parameter P will be equal to one for those bits. Parameter P will be equal to 0 for bits equal to 0 (which have been embedded in LSR B).

Finally, we need to have a new parameter to express the reliability of the retrieved signature. Let's name it by σ:

$$\sigma(\%) = \frac{\sum |P_i - \xi|}{l * \xi} * 100, i = 0,\ldots,l$$

where ξ is the threshold value and l is the length of the signature.

IV.1. Case n=4

Figure 5 shows, for a 'n' equal to 4, the original image 'Lena', the decoded image of 'Lena' with no signature, and the decoded image of 'Lena' with the signature. Both, the Peak to Signal Noise Ratio (PSNR) between original and signed image and that between original and non-signed image present a value higher than 31.5 dB.

(a) (b) (c)

Fig. 5 : (a) Original 'Lena' image; (b) decoded image of 'Lena' with no signature; (c) decoded image of 'Lena' with the signature (n=4)

We have tested the robustness against JPEG compression qualities of 90, 75 and 50 %. Table 1 shows the results:

	No JPEG	JPEG 90%	JPEG 75%	JPEG 50%
P mean (even bits)	0.985	0.672	0.457	0.351
P mean (odd bits)	0.00	0.099	0.18	0.219

Threshold ξ	0.5	0.39	0.34	0.31
Reliability σ (%)	98.5	73.5	40.6	21.4
Bits correctly retrieved	32	32	32	29

Table I

IV.2. Case n=8

Figure 6 shows the original image 'Lena', the decoded image of 'Lena' with no signature, and the decoded image of 'Lena' with the signature. The PSNR between the original image and the non signed image presents a value of 25.82 dB (eighth iteration) whereas that between the original one and the signed decoded is equal to 25.40 dB (eighth iteration).

(a) (b) (c)

Fig. 6 : (a) Original 'Lena' image; (b) decoded image of 'Lena' with no signature; (c) decoded image of 'Lena' with the signature (n=8)

As before, we show in next table the behavior against JPEG compression. The same rates of quality have been tested:

	No JPEG	JPEG 90%	JPEG 75%	JPEG 50%
P mean (even bits)	0.992	0.962	0.887	0.797
P mean (odd bits)	0.00	0.002	0.035	0.097
Threshold ξ	0.5	0.52	0.48	0.46
Reliability σ (%)	99.25	92.3	88.8	77.2
Bits correctly retrieved	32	32	32	32

Table II

Table III shows the results when we have performed the tests of Table II but with a previous blurring on the signed image.

	No JPEG	JPEG 90%	JPEG 75%
P mean (even bits)	0.662	0.642	0.567
P mean (odd bits)	0.105	0.085	0.145
Threshold ξ	0.40	0.36	0.36
Reliability σ (%)	69.7	77.4	62.1
Bits correctly retrieved	32	32	32

Table III

V. Conclusions and Future Works

We have presented an algorithm which succeeds in making digital signature functionality by modifying fractal features of the image.

The presented results show a good behavior for JPEG compression when n = 4. In that case, the algorithm has proved to be able to retrieve correctly the signature up to a JPEG quality of 75 % with a reliability of 40.8 %. When the quality went down to a value of 50 %, the number of badly recognized bits of the signature was of only three, though the reliability was, in that case, of 21.4 %.

When n = 8, the robustness against JPEG compression has turned out to be pretty high even for a quality of 50 % (reliability equal to 77.2 %). The method would have probably proved to be robust to higher compression rates though at these stages JPEG images may become damaged.

Concerning low pass filtering, the tests that have been performed for n = 4, showed some weakness against blurring convolutions. But for n = 8, the technique appeared to be very robust, even when the blurring attack was followed by a JPEG compression. When the compression rate was of 75%, we were able to retrieve the signature with a good reliability (σ = 62.1 %).

For the case n = 4, the quality of the decoded images may be sufficient for some applications. However, the low complexity of the technique suggests that this quality can still be improved. On the other hand, new improvements ought to combine both cases, n=4 and n=8, in order to get either a good robustness against JPEG compression and low pass filtering and an acceptable level of quality. A quadtree-based algorithm seems promising as a mean to achieve this compromise. Indeed, a more advanced version of the actual work should take into account the statistics of the blocks where the signature is embedded.

A feature of this technique is that it does not allow, once the image has been decoded, to find out the location of the signature (in fact, it does not allow to determine whether a signature has been embedded or not). Since a Local Iterated Function Systems based algorithm looks for a set of transformations able to give a good approximation to the image to encode, it does not matter what these transformations are if the approximation is good enough. What the algorithm does, in fact, is to distinguish between different set of transformations by giving to one of them the feature of constituting a signature. Related to the last point would be the fact that we let totally opened the choice of searching regions shapes. The optimization of these shapes might increase the robustness of the signature as well as the retrieving reliability.

We think also that the *Fractal Vector Technique (FVT)* might be suitable in systems where images broadcasting needs a rate of compression similar to the one given by the fractal coding. Experiences have not shown great differences in quality between the decoded images either containing a signature or not. In effect, the degradation of the images comes mainly from the nature of the fractal method, not from the introduction of a watermark.

Finally, fractal compression scheme extracts several other parameters that might also be used to sign the image.

VI. References

[1] J. Puate, F. Jordan, Two New Approaches to Digital Image Signature, *Diploma Project March 1996.*

[2] C. de Sola, F. Jordan, Enhancement of a watermarking algorithm in order to increase resistance to JPEG compression, *Diploma Project March 1995.*

[3] T. H. Kaskalis, I. Pitas, Applying Signatures on Digital Images.

[4] O. Bruyndonckx, J.-J Quisquater and B. Macq. Spatial Method for Copyright Labeling of Digital Images.

[5] Kineo Matsui, Kiyoshi Tanaka. Video-Steganography: How to Secretly Embed a Signature in a Picture.

[6] S. Burgett, E. Koch, S. Zhao, A Novel Method for Copyright Labeling Digitized Image Data.

[7] R. G. Van Schyndel. A Digital Watermark.

[8] Ingemar J. Cox, Joe Kilian, Tom Leighton, Talal Shamoon. Secure Spread Spectrum Watermarking for Multimedia. *NEC Research Institute, Technical Report 95 - 10.*

[9] T. Vynne, F. Jordan, Embedding a Digital Signature in a Video Sequence using Motion Vectors. *Submitted to ICIP 96.*

[10] M. F. Barnsley. *Iterated function systems.* In R. L. Devaney, L. Keen, K. T. Alligood, J. A. Yorke, M. F. Barnsley, B. Branner, J. Harrison, and P. J. Holmes, editors, Chaos and Fractals: The Mathematics Behind the Computer Graphics. American Mathematical Society, 1989.

[11] M. F. Barnsley, *Methods and apparatus for image compression by iterated function systems.* United States Patent Number 4, 941, 193, 1990.

[12] M. F. Barnsley and S. Demko. Iterated function systems and the global construction of fractals. *Proceedings of the Royal Society of London,* A399: 243-275, 1985

[13] M. F. Barnsley and L. P. HUD, *Fractal Image Compression.* AK Peters, Ltd., Wellesley, Massachusetts, 1993.

[14] Jacquin A., Image Coding Based on a fractal Theory of Iterated Contractive Image Transformations, *IEEE Transactions on image processing,* VolI, pp 18-30. January 1992.

[15] L. Torres, M. Kunt, Video Coding, The second Generation Approach.

[16] Fisher Y., *Fractal Image Compression: Theory and Application,* Spinger Verlag Edition, New York, 1995.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

693

6

Proceedings

INTERNATIONAL CONFERENCE ON IMAGE PROCESSING

September 16 – 19, 1996
Lausanne, Switzerland

Sponsored by
The IEEE Signal Processing Society

Volume III of III

TRANSPARENT ROBUST IMAGE WATERMARKING

Mitchell D. Swanson, Bin Zhu, and Ahmed H. Tewfik

Department of Electrical Engineering
University of Minnesota, Minneapolis, MN 55455 USA
email: mswanson, binzhu, tewfik@ee.umn.edu

ABSTRACT

We propose a watermarking scheme to hide copyright information in an image. The scheme employs visual masking to guarantee that the embedded watermark is invisible and to maximize the robustness of the hidden data. The watermark is constructed for arbitrary image blocks by filtering a pseudo-noise sequence (author id) with a filter that approximates the frequency masking characteristics of the visual system. The noise-like watermark is statistically invisible to deter unauthorized removal. Experimental results show that the watermark is robust to several distortions including white and colored noises, JPEG coding at different qualities, and cropping.

1. INTRODUCTION

Digital images facilitate efficient distribution, reproduction, and manipulation over networked information systems. However, these efficiencies also increase the problems associated with copyright enforcement. To address this issue, digital watermarks (i.e., author signatures) are under investigation. Watermarking is the process of encoding hidden copyright information in an image by making small modifications to its pixels. Unlike encryption, watermarking does not restrict access to an image. Watermarking is employed to provide solid proof of ownership. To be effective, the watermark must be [1, 2]: **perceptually invisible** within the host media; **statistically invisible** to thwart unauthorized removal; **readily extracted** by the image owner; and **robust** to incidental and intended signal distortions incurred by the host image, e.g., filtering, compression, re-sampling, re-touching, cropping, etc.

In this paper, we introduce a novel watermarking scheme for images which exploits the human visual system (HVS) to *guarantee that the embedded watermark is imperceptible*. Our watermark is generated by filtering a pseudo-noise sequence (author id) with a filter that approximates the frequency masking characteristics of the HVS. The image watermark is constructed by computing watermarks for individual image blocks. The blocks may be $n \times m$ or may be defined in terms of image objects/regions. This helps deter pirating of image objects. Furthermore, the noise-like watermark is statistically invisible. We include experimental results which indicate that the watermark is readily extracted and *robust* to common signal processing operations.

This work was supported by AFOSR under grant AF/F49620-94-1-0461.

2. PREVIOUS WORK

The most common watermarking approaches modify the least significant bits (LSB) of an image based on the assumption that the LSB data are insignificant. Two LSB techniques are described in [3]. The first replaces the LSB of the image with a pseudo-noise (PN) sequence, while the second adds a PN sequence to the LSB of the data. Another LSB data hiding method called "Patchwork" [1] chooses n pairs (a_i, b_i) of points in an image and increases the brightness of a_i by one unit while simultaneously decreasing the brightness of b_i. Several executable software packages (e.g., Stego, S-Tools) based on LSB approaches are also available. However, any approach which only modifies the LSB data is *highly sensitive to noise* and is easily destroyed. Furthermore, image quality may be degraded by the watermark. Other watermarking approaches include [4, 5, 6].

A method similar to ours is presented in [7], where the authors hide data by adding fixed amplitude pseudo-noise to the image. The approach presented here employs masking to vary the amplitude of the hidden data. Specifically, the tolerable error levels obtained using masking provide us with the maximum amount the image data may change. Pseudo-noise techniques are also used in [2], where the N largest frequency components of an image are modified by Gaussian noise. However, the scheme only modifies a subset of the frequency components and does not take into account the HVS. The watermark we propose here embeds the max-

0-7803-3258-X/96/$5.00 © 1996 IEEE

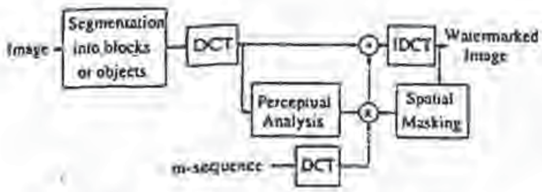


Figure 1: Diagram of new watermarking technique.

imum amount of information throughout the spectrum. Since more data is embedded, this scheme is *guaranteed* to be more robust to modifications than a technique which only modifies a subset of the image data.

3. WATERMARK GENERATION

In Fig. 1, we show our watermarking technique. The initial step consists of segmenting the image into blocks. Using a traditional approach, the blocks may be $n \times n$ (e.g., 8×8 like JPEG). An option at this stage is to segment the image into blocks of *objects and texture regions*. In either case, blocking the image adds detection robustness to cropping and localized signal processing operations. Upon applying a discrete cosine transform (DCT) to each block, a frequency mask is computed for each block in a manner similar to low bit rate coding algorithms [8]. The resulting perceptual mask is scaled and multiplied by the DCT of a maximal length pseudo-noise sequence (author id). Note that a different pseudo-noise sequence is used for each image block. This *watermark* is then added to the corresponding DCT block. The watermarked image is obtained by assembling the inverse DCT's of each block. Spatial masking is used to verify that the watermark is invisible and to control the scaling factor.

Pseudo-noise (PN) sequences form the signatures in our watermarking scheme because of their noise-like characteristics, resistance to interference, and their good auto-correlation properties. PN sequences are periodic noise-like binary sequences generated by length m linear shift registers [9]. Furthermore, the period N autocorrelation function has peaks equal to 1 at 0, N, 2N, etc., and is approximately equal to $1/N$, elsewhere. These periodic peaks allow the author to *synchronize* with the embedded watermark during the detection process.

Visual masking models are used to modify the author signature. Visual masking refers to a situation where a signal raises the visual threshold for other signals around it. Both frequency and spatial masking are employed by our watermarking scheme. Our frequency masking model is based on the observation that a masking grating raises the visual threshold for signal gratings around the masking frequency [10]. The model we use [11] expresses the contrast threshold at frequency f as a function of f, the masking frequency f_m and the masking contrast c_m:

$$c(f, f_m) = c_0(f) \cdot Max\{1, [k(f/f_m)c_m]^{\alpha}\},$$

where $c_0(f)$ is the detection threshold at frequency f. To find the contrast threshold $c(f)$ at a frequency f in an image, we first use the DCT to transform the image into the frequency domain and find the contrast at each frequency. Then we use a summation rule of the form $c(f) = [\sum_{f_m} c(f, f_m)^{\beta}]^{1/\beta}$. If the contrast error at f is less than $c(f)$, the model predicts that the error is invisible to human eyes.

After adding the watermark in the frequency domain, spatial masking is checked. The spatial model is used to verify that the watermark designed with the frequency masking model is invisible for local spatial regions. The model used here is similar to our image coding model [11] which gives the tolerable error level for each coefficient. Each watermark coefficient is compared with the tolerable error level obtained to assure that it is invisible. A visible watermark is rescaled via a weighting factor.

4. WATERMARK DETECTION

The watermark should be extractable even if common signal processing operations are applied to the host image. This is particularly true in the case of deliberate unauthorized attempts to remove it. For example, a pirate may attempt to add noise, filter, code, re-scale, etc., an image in an attempt to destroy the watermark. As the embedded watermark is noise-like and its location (based on multiple blocks) is unknown, a pirate has insufficient knowledge to directly remove the watermark. Furthermore, a different m-sequence is used for each block to further reduce unauthorized watermark removal by cross-correlation. Therefore, any destruction attempts are done blindly. Unlike other users, the author has copies of the original signal S and the signature. Detection of the watermark is accomplished via hypotheses testing:

$$\begin{array}{lll} H_0: & X = R - S = N & \text{(No watermark)} \\ H_1: & X = R - S = W' + N & \text{(Watermark)} \end{array}$$

where R is the potentially pirated signal, W' is the potentially modified watermark, and N is noise. The correct hypothesis is obtained by applying a correlating detector on X with W and comparing with a threshold. In some cases, e.g., spatial rescaling, a generalized likelihood ratio test must be applied.

5. EXPERIMENTAL RESULTS

To illustrate our watermarking technique, the 256×256 grayscale (8-bit) image shown in Fig. 2 was segmented into 8×8 blocks and watermarked. The watermarked image is shown in Fig. 3. The images appear identical.

We tested the robustness of the watermark to several degradations. To model perceptual coding techniques, we corrupted the watermark with *worst case colored noise which follows the image mask*. We generated colored noise with SNR of 10dB and added it to the image with (hypothesis H_1) and without (H_0) the watermark. The watermarked image with colored noise is shown in Fig. 4. The hypothesis test was applied to each block in the image. This testing process was repeated 250 times. The normalized correlation coefficients indicate easy discrimination between the hypotheses as shown in Fig. 6. In particular, the correlation coefficient for the image with and without the watermark was approximately 0 and 1 respectively.

To further degrade the watermark, we applied JPEG coding at 0.38 bpp (10% quality, c.f. Fig. 5) and 1.32 bpp (50% quality) to each of the images *already corrupted* with colored noise. Note that the image is significantly degraded at 0.38 bpp, yet the watermark is still easily detected as shown in Fig. 6. It is unlikely a pirate would do so much irreparable damage to the image. Setting a decision threshold of 0.15 results in no decision errors. In Fig. 7, we show the result of applying JPEG coding at different quality factors to the noisy image with and without the watermark. It is clear that the correlation coefficient values for the two hypotheses are well separated for all JPEG coding qualities. We also investigated cropping robustness by determining the minimum number of image blocks required to make a confident decision on whether the watermark is present in an image ($P_D = 1$ and $P_F < 10^{-4}$). Each noisy image in the above tests was *randomly cropped* and tested. The results indicate that only 0.4%, 2%, and 15% of the image is needed for a confident decision when coded at 8 bpp, 1.32 bpp, and 0.38 bpp.

For comparison, we implemented the system described in [2]. For JPEG coding at 10%, we obtained (unnormalized) correlation coefficients using their system of 5.09 and 2.34 for the same test image with and without the watermark, respectively. The ratio is significantly smaller than ours. While testing their system, we were unable to reproduce the detection results as claimed in [2]. This may be the result of special post-processing operations they implement. The robustness of our watermarking scheme to re-sampling, multiple watermarking, vector quantization, and other distortions is described in [12].

6. REFERENCES

[1] W. Bender, D. Gruhl, and N. Morimoto, "Techniques for Data Hiding." Tech. Rep., MIT Media Lab, 1994.

[2] I. Cox, J. Kilian, T. Leighton, and T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia." Tech. Rep. 95-10, NEC Research Institute, 1995.

[3] R. G. van Schyndel, A. Z. Tirkel, and C. F. Osborne, "A Digital Watermark," in *Proc. 1994 IEEE Int. Conf. on Image Proc.*, vol. II, (Austin, TX), pp. 86–90, 1994.

[4] I. Pitas and T. Kaskalis, "Applying signatures on digital images," in *Nonlinear Signal Processing Workshop, Thessaloniki, Greece*, pp. 460–463, 1995.

[5] E. Koch and J. Zhao, "Towards robust and hidden image copyright labeling," in *Nonlinear Signal Processing Workshop, Thessaloniki, Greece*, pp. 452–455, 1995.

[6] O. Bruyndonckx, J.-J. Quisquater, and B. Macq, "Spatial method for copyright labeling of digital images," in *Nonlinear Signal Processing Workshop, Thessaloniki, Greece*, pp. 456–459, 1995.

[7] J. R. Smith and B. O. Comiskey, "Modulation and Information Hiding in Images." to appear *1996 Workshop on Information Hiding*, University of Cambridge, UK.

[8] N. Jayant, J. Johnston, and R. Safranek, "Signal Compression Based on Models of Human Perception," *Proc. of the IEEE*, vol. 81, pp. 1385-1422, oct 1993.

[9] S. Haykin, *Communication Systems, 3rd Edition*. New York, NY: John Wiley and Sons, 1994.

[10] G. E. Legge and J. M. Foley, "Contrast Masking in Human Vision," *J. Opt. Soc. Am.*, vol. 70, no. 12, pp. 1458-1471, 1980.

[11] B. Zhu, A. Tewfik, and O. Gerek, "Low Bit Rate Near-Transparent Image Coding," in *Proc. of the SPIE Int. Conf. on Wavelet Apps. for Dual Use*, vol. 2491, (Orlando, FL), pp. 173–184, 1995.

[12] A. H. Tewfik, M. D. Swanson, B. Zhu, K. Hamdy, and L. Boney, "Transparent Robust Watermarking for Images and Audio." To be submitted *IEEE Trans. on Signal Proc.*, 1996.

Figure 2: Original 256x256 grayscale image.

Figure 3: Watermarked image using 8 × 8 blocks.

Figure 4: Watermarked image with colored noise (SNR 10dB).

Figure 5: JPEG coded version of watermarked image at 0.38bpp (10% quality).

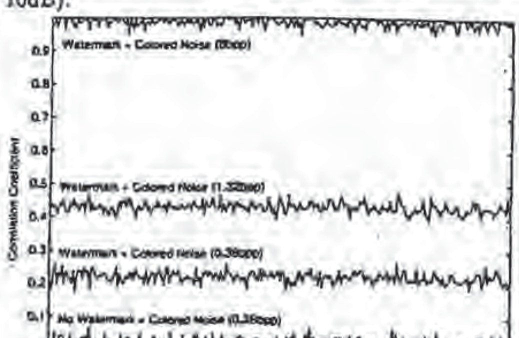


Figure 6: Watermark detection after adding colored noise with and without JPEG coding.

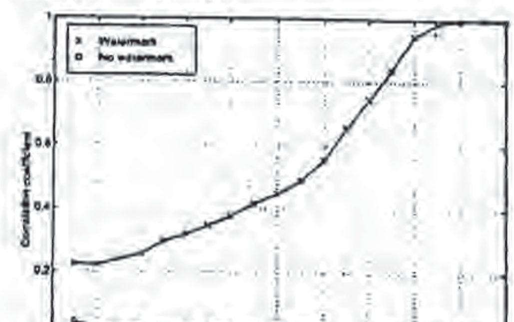


Figure 7: Watermark detection after JPEG coding at different qualities.

BEST AVAILABLE COPY

Signat
impor
inforn
ing di
effecti
in thi
digita
to sut
provic
topics

The f
ing a
ing d
"digi
impo

1.

2.

3.

4.

The
vide
on i
pers
gin

0-78

692

7

1996 IEEE DIGITAL SIGNAL PROCESSING WORKSHOP PROCEEDINGS

SEPTEMBER 1 – 4 , 1996

HOTEL ALEXANDRA, LOEN, NORWAY

IEEE

NORSIG

Cosponsored by: ABB Technology NERA Telecommunications

Robust Data Hiding for Images *

Mitchell D. Swanson *Bin Zhu* *Ahmed H. Tewfik*
Department of Electrical Engineering, University of Minnesota, Minneapolis, MN 55455
e-mail: mswanson, binzhu, tewfik@ee.umn.edu

ABSTRACT

Data hiding is the process of encoding extra information in an image by making small modifications to its pixels. To be practical, the hidden data must be perceptually invisible yet robust to common signal processing operations. This paper introduces two schemes for hiding data in images. The techniques exploit perceptual masking properties to embed the data in an invisible manner. The first method employs spatial masking and data spreading to hide information by modifying image coefficients. The second method uses frequency masking to modify image spectral components. By using perceptual masking, we also increase robustness of the hidden information. Experimental results of data recovery after applying noise and JPEG coding to the hidden data are included.

1. INTRODUCTION

We introduce two robust schemes to hide information, e.g., labels, into an image by modifying perceptually irrelevant portions of image data. By exploiting the human visual system (HVS), our techniques embed a large amount of data into an image while guaranteeing that the hidden data are perceptually invisible. In particular, the data are hidden by modifying image coefficients according to masking levels based on the HVS. Information may be hidden throughout the image or confined to particular image objects and regions. The first scheme we introduce spreads the data to be hidden with a pseudo-noise sequence and then modifies them using spatial masking. The second data hiding scheme modifies the DCT coefficients of image blocks according to their frequency masking characteristics. In both schemes, masking characteristics are used to maintain high bit rates and robustness of the hidden data. We include experimental results which indicate the robustness of the data hiding techniques to common signal processing operations.

To be useful, the embedded data must be [1, 2]: *perceptually invisible* within the host media; *readily extracted* by its intended audience; *high bit-rate* for practical applications; and *robust* to manipulation and signal processing operations on the host image, e.g., filtering, re-sampling, compression, noise, cropping, etc. Hiding information in images may be used, e.g., to supplement an image with additional information, add copyright protection (i.e., digital watermarks), or verify image integrity. We consider here the generic problem of embedding supplementary information into image data with the goal of recovering the information bits *without* access to the original image. In such a scenario, the author of the hidden information supplies a public key which allows users to recover the hidden data. The hidden information may be text, audio, or image data. For example, text captions may be used to label faces and buildings in an image. A short audio clip may associate a train whistle with an image of a locomotive.

Note that embedding information *directly into image data* has several advantages over storing the data in an image header or a separate file. Specifically, header data are easily lost when the format of a file is changed or the image is cropped. Separate files also need to be transmitted when the image is transmitted and are difficult to maintain during cropping operations. Separate files and image headers also require additional storage. Hiding data directly into the image data resolves these problems.

2. PREVIOUS WORK

Several data hiding techniques have been proposed. The most common approaches modify the least significant bits (LSB) of an image based on the assumption that the LSB data are insignificant. In [3], two such techniques are described. The first replaces the LSB of the image with a pseudo-noise (PN) sequence, while the second adds a PN sequence to the LSB of the data. However, any approach which only modifies the LSB data is *highly sensitive to noise* and is easily destroyed. Another LSB data hiding method called "Patchwork" [1] chooses n pairs (a_i, b_i) of points in an image and increases the brightness of a_i by one unit while simultaneously decreasing the brightness of b_i. Several executable software packages (e.g., Stego, S-Tools) based on LSB approaches are also available [4]. A data hiding method similar to our frequency hiding method is proposed in [2], where the N largest frequency components of an image are modified by a PN sequence. How-

*This work was supported by AFOSR under grant AF/F49620-94-1-0461.

0-7803-3629-1/96/$5.00 © 1996 IEEE

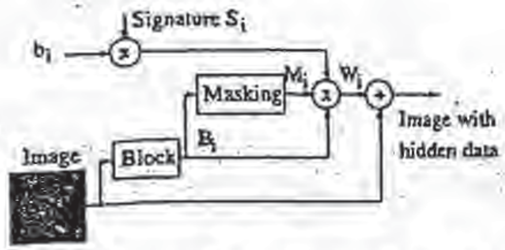


Figure 1. Diagram of spread spectrum data hiding technique.

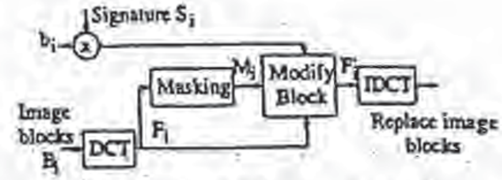


Figure 2. Diagram of frequency data hiding technique.

ever, the scheme only modifies a subset of the frequency components and does not take into account the HVS. A method similar to our spatial hiding scheme is presented in [5], where the authors hide data in an image using spread spectrum techniques. Our approach employs spatial masking to *maximize* the amount of data hidden in the image.

3. SPATIAL AND FREQUENCY DOMAIN MASKING

We use masking models based on the HVS to ensure that the hidden data are perceptually invisible. Visual masking refers to a situation where a signal raises the visual threshold for other signals around it. Masking characteristics are used in high quality low bit rate coding algorithms to further reduce bit rates [6].

Our frequency masking model is based on the knowledge that a masking grating raises the visual threshold for signal gratings around the masking frequency [7]. The model we use [8] expresses the contrast threshold at frequency f as a function of f, the masking frequency f_m and the masking contrast c_m. To find the contrast threshold $c(f)$ at a frequency f in an image, we first use the DCT to transform the image into the frequency domain and find the contrast at each frequency. If the contrast error at f is less than $c(f)$, the model predicts that the error is invisible to human eyes.

Spatial masking refers to the situation that an edge raises the perceptual threshold around it. The model used here is similar to our image coding model [8] which is based on a model proposed by Girod [9]. In our approach, the upper channel of Girod's model is linearized under the assumption of small perceptual errors. The model gives the tolerable error level for each pixel in the image.

4. SPATIAL DATA HIDING

In Fig. 1, we show our first data hiding technique which uses spatial masking to shape hidden data. The first step consists of selecting arbitrarily sized image blocks B_i to embed the data. Note that the data may be embedded throughout the image or localized to specific regions and objects (e.g., the face outlined with a black box). This allows the author to associate image features with specific captions.

The message bits b_i are first spread using a pseudo-noise (PN) author signature S_i. As with spread spectrum communication systems, the PN sequence spreads the data spectrum, increases noise resistance, and hides the data. Spatial masking is then computed for the image blocks. The masking blocks M_i (i.e., tolerable error levels) are used to modify the hidden data $W_i = b_i(B_i * M_i * S_i)$, where $*$ is element-wise multiplication. The hidden data W_i are then added to the image. The masking blocks M_i shape the hidden data to guarantee invisibility and increase robustness.

As the hidden data are noise-like, they may only be extracted by a receiver that knows the PN signature S_i. A conventional receiver with access to S_i may be used to detect the hidden data. Specifically, the received image data blocks B_i' are projected onto the *author signature* S_i *weighted by the estimated image mask* M_i' and then thresholded to obtain an estimate of the data bit. In this way, image coefficients which are modified the most are given the most weight when the hidden data is recovered. Note that if the image has been cropped or translated, signature synchronization can be obtained using a two dimensional search. In some cases, e.g., spatial rescaling, a generalized likelihood ratio test must be applied.

5. FREQUENCY DATA HIDING

Our second data hiding approach is based on the frequency masking characteristics of the image data. The scheme is shown in Fig. 2. Again, blocks B_i are selected to hide the data b_i which are first spread by signature S_i. A discrete cosine transform (DCT) is applied to each block to form a DCT block F_i. A perceptual analysis stage is then applied to each DCT block to form frequency masking blocks M_i. A bit b_i is hidden in block F_i by modifying the DCT coefficients according to the equation

$$F_i'(j,k) = \left(\left\lfloor \frac{F_i(j,k)}{M_i(j,k)} \right\rceil + \frac{1}{4} b_i S_i(j,k)\right) M_i(j,k),$$

where $\lfloor\cdot\rceil$ denotes the rounding operation. The original image blocks B_i are replaced by the inverse DCT's of the modified blocks F_i'. Spatial masking is applied to the modified image to verify that the hidden data are invisible.

Given the image with (possibly modified) hidden data blocks F_i'', the data bit b_i may be recovered by forming the difference

$$\hat{b}_i = \sum_{j,k} M_i^{\prime}(j,k) sgn\left(\frac{F_i^{\prime\prime}(j,k)}{M_i^{\prime}(j,k)} - \left[\frac{F_i^{\prime\prime}(j,k)}{M_i^{\prime}(j,k)}\right]\right)$$

where $M_i^{\prime}$ is the frequency mask *estimated* by the receiver times the signature S_i, i.e., $M_i^{\prime} = M_i^{est} * S_i$, and $sgn(\cdot)$ is the sign value. Again, the bit decision for block B_i is weighted by the mask $M_i^{\prime}$. Unlike spatial data hiding, the bit error rate (BER) of this scheme is *zero* when no distortion is present in the received image. We can derive a simple expression for the upper bound on the BER when zero mean Gaussian noise with variance σ^2 is added to the signal. Without loss of generality, assume that $b_i = 1$. A decision error occurs for coefficient $F^{\prime\prime}(j,k)$ whenever the magnitude of a noise sample $|w(j,k)|$ falls in one of the intervals

$$\left[\frac{(4n+1)M(j,k)}{4}, \frac{(4n+3)M(j,k)}{4}\right] = I_n$$

for $n = 0, 1, 2, \ldots$. Using the complementary error function $erfc(\cdot)$, we may write the probability of error for coefficient $F^{\prime\prime}(j,k)$ as

$$P_e(F^{\prime\prime}(j,k), \sigma) = 2\sum_{n=0}^{\infty} erfc(\frac{I_n}{\sigma}).$$

For σ fixed, $P_e(F^{\prime\prime}(j,k), \sigma)$ decreases as $M(j,k)$ increases. Hence the receiver places more weight on coefficients with large masking values. The overall probability of error for bit b_i is a weighted combination of the $P_e(F^{\prime\prime}(j,k), \sigma)$ in block B_i.

6. PREPROCESSING HIDDEN DATA

To add further robustness to the hidden data, the data hiding techniques introduced here may be modified to take into account certain signal processing operations. If it is known that a JPEG coder will be applied to the image, for example, we can modify each data hiding procedure appropriately. In the spatial data hiding case, the signature-masking product is computed and then compressed/decompressed using a JPEG coder at an estimated quality factor Q. The decompressed product can then be modified to compensate for energy losses. In the frequency hiding scheme, the mask M_i may be preprocessed using the JPEG quantization table by substituting a new mask $\hat{M}_i = Q * M_i$ for M_i.

7. RESULTS

We illustrate our data hiding techniques on the 256×256 grayscale image shown in Fig. 3. Using each scheme, we embedded the text "We are investigating data hiding in multimedia systems" (432 bits) in the image by converting the text into bits and embedding each bit in an 8×8 image block. The text "Lena, 123 Main Street" (168 bits) is also hidden in blocks about the face object in the image (outlined by the block box in Fig. 1). The image with the hidden data is shown in Fig. 4.

In Fig. 5 we show a plot of BER for differing levels of white noises using spatial data hiding. Note that since the signature and noise are approximately uncorrelated, the BER remains constant for all noise levels. The BER of the scheme for JPEG coding at different quality settings is shown in Fig. 6. Preprocessing of the signature-mask product at a quality of 80% was used.

A plot of BER for differing levels of white noises using frequency data hiding is shown in Fig. 7. Note that this scheme performs better in low noise conditions and worse in high noise conditions than the spatial technique. Similarly for the plot of BER versus JPEG coding at different quality settings shown in Fig. 8. The frequency scheme works well under high quality coding conditions yet degrades more rapidly than spatial data hiding when the coding becomes too lossy. JPEG preprocessing at a quality of 70% was used.

REFERENCES

[1] W. Bender, D. Gruhl, and N. Morimoto, "Techniques for Data Hiding." Tech. Rep., MIT Media Lab, 1994.

[2] I. Cox, J. Kilian, T. Leighton, and T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia." Tech. Rep. 95-10, NEC Research Institute, 1995.

[3] R. G. van Schyndel, A. Z. Tirkel, and C. F. Osborne, "A Digital Watermark," in *Proc. 1994 IEEE Int. Conf. on Image Proc.*, vol. II, (Austin, TX), pp. 86–90, 1994.

[4] E. Milbrandt, "Steganography Info and Archive." Available at URL *http://indyunix.iupui.edu/~emilbran/stego.html*.

[5] J. R. Smith and B. O. Comiskey, "Modulation and Information Hiding in Images." to appear *1996 Workshop on Information Hiding*, University of Cambridge, UK.

[6] N. Jayant, J. Johnston, and R. Safranek, "Signal Compression Based on Models of Human Perception," *Proc. of the IEEE*, vol. 81, pp. 1385–1422, oct 1993.

[7] G. E. Legge and J. M. Foley, "Contrast Masking in Human Vision," *J. Opt. Soc. Am.*, vol. 70, no. 12, pp. 1458–1471, 1980.

[8] B. Zhu, A. Tewfik, and O. Gerek, "Low Bit Rate Near-Transparent Image Coding," in *Proc. of the SPIE Int. Conf. on Wavelet Apps. for Dual Use*, vol. 2491, (Orlando, FL), pp. 173–184, 1995.

[9] B. Girod, "The Information Theoretical Significance of Spatial and Temporal Masking in Video Signals," in *Proc. of the SPIE Human Vision, Visual Processing, and Digital Display*, vol. 1077, pp. 178–187, 1989.

Figure 3. Original 256x256 grayscale image.

Figure 4. Image with hidden data.

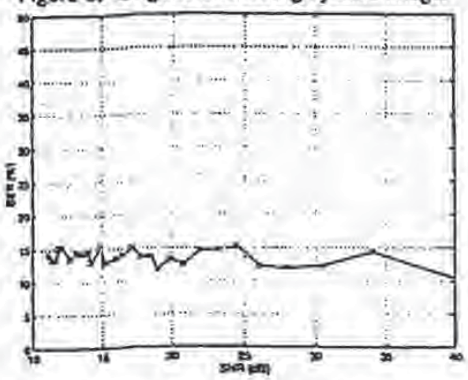


Figure 5. Bit error rate versus SNR using spatial data hiding.

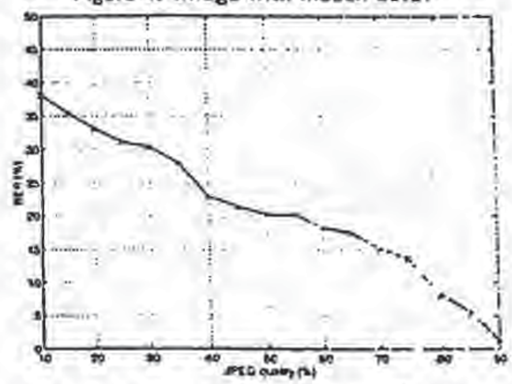


Figure 6. Bit error rate versus JPEG coding at different qualities using spatial data hiding.

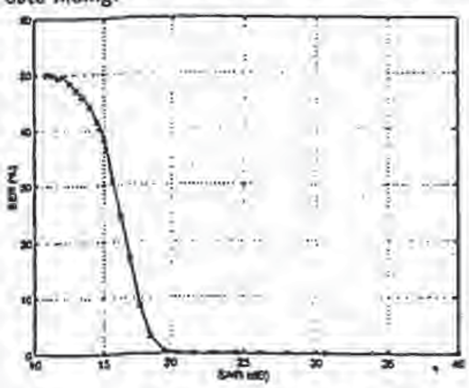


Figure 7. Bit error rate versus SNR using frequency data hiding.

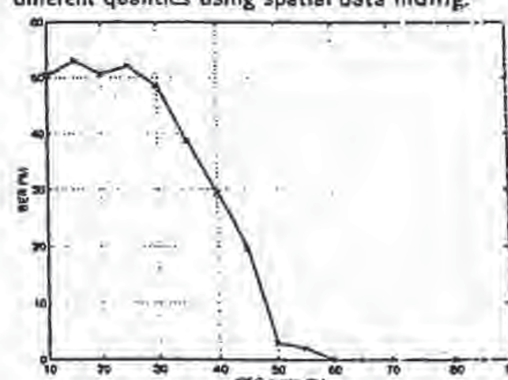


Figure 8. Bit error rate versus JPEG coding at different qualities using frequency data hiding

Klaus Brunnstein, Peter Paul Sint

Intellectual Property Rights and New Technologies

Proceedings of the KnowRight'95 Conference

R.Oldenbourg Wien München 1995

EMBEDDING ROBUST LABELS INTO IMAGES FOR COPYRIGHT PROTECTION

Jian Zhao & Eckhard Koch
Fraunhofer Institute for Computer Graphics
Wilhelminenstr. 7, 64283 Darmstadt, Germany
Email: {zhao, ekoch}@igd.fhg.de

Abstract

This paper describes a set of novel steganographic methods to secretly embed robust labels into image data for identifying image copyright holder and original distributor in digital networked environment. The embedded label is undetectable, unremovable and unalterable. Furthermore it can survive processing which does not seriously reduce the quality of the image, such as lossy image compression, low pass filtering and image format conversions.

1 Introduction

The wide use of digitally formatted audio, video and printed information in network environment has been slowed down by the lack of adequate protection on them. Developers and publishers hesitate to distribute their sensitive or valuable materials because of the easiness of illicit copying and dissemination [3],[6],[7].

Compared to ordinary paper form information, digitized multimedia information (image, text, audio, video) provides many advantages, such as easy and inexpensive duplication and re-use, less expensive and more flexible transmission either electronically (e.g. through the Internet) or physically (e.g. as CD-ROM). Furthermore, transferring such information electronically through network is faster and needs less efforts than physical paper copying, distribution and update. However, these advantages also significantly increase the problems associated with enforcing *copyright* on the electronic information.

Basically, in order to protect distributed electronic multimedia information, we need two types of protections. First, the multimedia data must contain a label or code, which identifies it uniquely as property of the copyright holder. Second, the multimedia data should be marked in a manner which allows its distribution to be tracked. This does not limit the number of copies allowed (vs. copy protection), but provides a mean to check the original distributor. In order to prevent any copyright forgery, misuse and violation, the copyright label must be unremovable and unalterable, and furthermore survive processing which does not seriously reduce the quality of the data. This requires that first the label must be secretly stored in a multimedia data, i.e. the locations for embedding this label are secret, second the label must be robust even if the labeled multimedia data has been processed incidentally or intentionally.

This paper describes a set of novel steganographic methods to secretly embed robust labels into image data for copyright protection in open networked environment. The label embedded in the image can be assigned or generated in a way that it is able to identify the copyright holder and the original purchaser (distributor).

Steganographic method is a technique embedding additional information into a data by modifying the original data without affecting the quality of the data. Many steganographic methods have been proposed to aim at storing additional information to identify or label formatted electronic documents [1], images, video [8], and audio data. However, they are far away from the requirements in protecting multimedia information in a networked environment, because although some of them provide secret locations for label embedding, none of them is able to prevent attacks on the embedded information by simple image processing, i.e. they do not adequately address the possibilities of using data compression, low pass filtering and/or simply changing the file format to remove an embedded code.

The discussion begins with a general framework for copyright label embedding. Then two specific methods are developed: one is based on the JPEG compression model for embedding labels in gray-scaled and color images, and the other is based on the black/white rate for binary images. Finally, these methods are tested experimentally and the future work is discussed.

2 Robust Label Embedding Framework

The system developed along the methods presented in this paper is called 'SysCoP' (System for Copyright Protection). It consists of a set of methods to embed robust labels into different types of images. Currently, the system supports gray-scaled, color, and binary images. These methods share an algorithm framework for both label writing and reading described below.

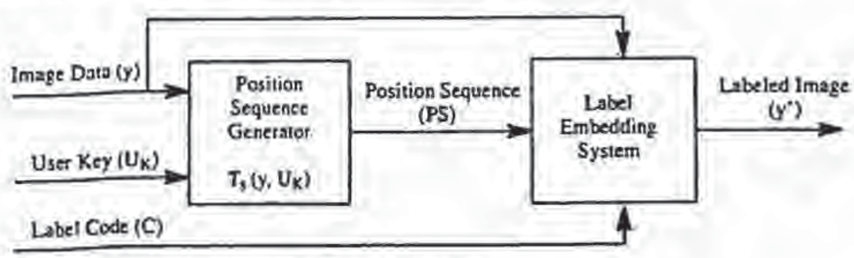


Figure 1. Write label

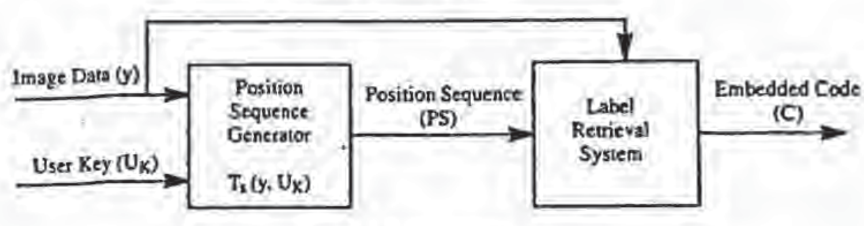


Figure 2. Read label

The framework, as shown in Figure 1 for label writing and Figure 2 for label reading, is composed of two steps. The first step generates a pseudo random position sequence for selecting blocks where the code is embedded. This step is denoted as a function $T_s(y, U_K)$ where y is the image data to be labeled, and U_K is the user-supplied secret key. The second step simply embeds or retrieves the code into or from the blocks specified in the position sequence. The methods for embedding or reading code depend on types of images, and will be described individually in the next section.

The function $T_s(y, U_K)$ firstly extracts some features from the image data and then use them together with the user secret key as the seeds for position sequence generation [4]. Ideally, the features of the image data used here must meet the following requirements:

- they must be robust against simple image processing that does not affect the visual quality of the image, and
- they must be image-dependent, i.e. the image can be identified, uniquely in an ideal case, by these features extracted from the image data.

However, to achieve the contradictory requirements above, in fact, is a very hard work. Much research has been done in related fields for other purposes, e.g. content-based image retrieval, image segment and pattern recognition, which can be found in many literature (e.g. [9]). Currently, we only use the width and height of an image for the position generation in the function $T_s(y, U_K)$.

Before describing the framework, we introduce some terminology. A *block* of an image consists of 8x8 pixels. In this framework, it is either a contiguous or a distributed block. A *contiguous block* is a 8x8 square in an image component. A *distributed block* is a collection of 8x8 pixels each of which is selected randomly from the whole image space, The purpose of distributed block is to prevent or discourage attackers from detecting embedding locations by comparing different labeled images. A block is 'invalid' for code embedding if too big modifications to the block data are needed in order to embed a bit into this block. The criteria of validation of the block depends on the specific label-embedding methods to be described in the next section.

Let C be the embedded code, and represented as a binary bit stream $\{c_0, c_1, ..., c_n\}$. Let i be the index of current bit in this stream. Let $\mathfrak{B}$ be the block set in which each block has been randomly selected. Initialize i to 0 and $\mathfrak{B}$ to { }. The framework for writing and reading robust labels is described below in Algorithm 1(a)-(b).

Algorithm 1(a): Framework (write).

(1) If $i \geq n$, return.

(2) Randomly select a block *b*, using the position sequence generation function $T_s(U_K, y)$ in Figure 1.

(3) If *b* exists already in $\mathfrak{B}$, go to (2), otherwise add *b* to $\mathfrak{B}$.

(4) Call *check_write(b, c_i)* to check wether *b* is a valid block: if this function returns False (i.e. the block *b* is an invalid block), go to (2).

(5) Call *write(b,c_i)* to embed a bit c_i to the block *b*.

(6) Increment i, go to (1).

Algorithm 1(b): Framework (read).

(1) If $i \geq n$, return.

(2) Randomly select a distributed or a contiguous 8x8 block b, using the position sequence generation function $T_s(U_K, y)$ in Figure 2.

(3) If b exists already in $\mathfrak{B}$, then go to (2), otherwise add b to $\mathfrak{B}$.

(4) Call ***check_read(b,c_i)*** to check wether b is a valid block: if this function returns False (i.e. the block b is an invalid block), go to (2).

(5) Call ***read(b)*** to retrieve a bit from the block b.

(6) Increment i, and go to (1).

3 Robust Label Embedding Methods

3.1 JPEG-Based Label Embedding for Gray-Scaled and Color Images

In this subsection, we first introduce briefly the JPEG compression model, then describe the principle of the embedding methods based on the JPEG compression model. Finally, the algorithms for embedding labels into gray-scaled and color images are developed.

Suppose the source image composes three components: one luminance (Y) and two chrominance (I and Q). That is, each pixel in the image can be represented with a triple of 8-bit values (Y,I,Q). Each component is broken up into contiguous blocks. The JPEG compression consists of six steps: normalization, DCT transformation, quantization, zigzag scan, run-length encoding and Huffman coding steps. Since our method is applied after the quantization step, we only describe briefly the first three steps of the JPEG model. The detailed description of the JPEG model is available elsewhere [11].

The normalization step brings all image values into a range, e.g. between −128 and 127 for a 24-bit image. The DCT step applies the discrete cosine transform (DCT) to each 8x8 block, producing a new 8x8 block [10]. If we call the new block Y(k,l), with $k,l \in 0..7$, the equation of the DCT is:

$$Y(k,l) = \frac{1}{4}\sum_i \sum_j C(i,k)C(j,l)y(i,j)$$

where

$$C(i,k) = A(k)\frac{\cos(2i+1)k\pi}{16} \qquad A(k) = \frac{1}{\sqrt{2}} \text{ for } k = 0,\ A(k) = 1 \text{ for } k \neq 0 \qquad (1)$$

Each element of the new block is further quantized:

$$Y_Q[k,l] = \mathrm{Round}\left(\frac{Y[k,l]}{q[k,l]}\right) \qquad (2)$$

Equation (2) represents the entire lossy modelling process of the JPEG compression. The choice of the quantization table (q[k,l]) determines both the amount of compression and the quality of the decompressed image. The JPEG standard includes recommended luminance and chrominance quantization tables resulting from human factors studies. To obtain different compression quality, we typically use a *quality factor* to scale the values of these default quantization tables.

In the JPEG decompression process, each element of $Y_Q(k,l)$ is multiplied by $q(k,l)$ to recover an approximation of $Y(k,l)$. Finally, the image block $y(i,j)$ can be recovered by performing an inverse 2-D DCT (IDCT):

$$y(i,j) = \frac{1}{4}\sum_k \sum_l C(i,k)C(j,l)Y(k,l) \qquad (3)$$

The basic principle of the JPEG-based embedding method is that quantized elements have a moderate variance level in the middle frequency coefficient ranges, where scattered changes in the image data should not be noticeably visible. The specific frequencies being used to embed the code will be 'hopped' in this range to increase the robustness of the signal and making it more difficult to find [5],[2]. A label bit is embedded through holding specific relationship among three quantized elements of a block. The relationships among them compose 8 patterns (combinations) which are divided into three groups: two of them are used to represent '1' or '0' for embedded codes (valid patterns), and the other represents *invalid patterns*. If too big modifications are needed to hold a desired valid pattern representing a bit, this block is invalid. In this case, the relationships among the three elements of the selected location set are modified to any of the invalid patterns to 'tell' the label-retrieval process that this block is invalid. The criterion for invalid blocks is specified by a parameter MD, i.e. the maximum difference between any two elements of a selected location set in order to reach the desired valid pattern.

Set No.	(k_1,l_1)	(k_2,l_2)	(k_3,l_3)
1	2(0,2)	9(1,1)	10(1,2)
2	9(1,1)	2(0,2)	10(1,2)
3	3(0,3)	10(1,2)	11(1,3)
4	10(1,2)	3(0,3)	11(1,3)
5	9(1,1)	2(0,2)	10(1,2)
6	2(0,2)	9(1,1)	10(1,2)
7	9(1,1)	16(2,0)	2(0,2)
8	16(2,0)	9(1,1)	2(0,2)
9	2(0,2)	9(1,1)	16(2,0)
10	9(1,1)	2(0,2)	16(2,0)
11	10(1,2)	17(2,1)	3(0,3)
12	17(2,1)	10(1,2)	3(0,3)
13	10(1,2)	3(0,3)	17(2,1)
14	3(0,3)	10(1,2)	17(2,1)
15	9(1,1)	16(2,0)	17(2,1)
16	16(2,0)	9(1,1)	17(2,1)
17	10(1,2)	17(2,1)	18(2,2)
18	17(2,1)	10(1,2)	18(2,2)

Table 1. Possible location sets

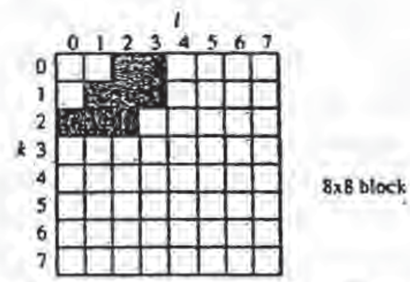


Figure 3. Possible locations for embedding code in a block

(k_1,l_1)	(k_2,l_2)	(k_3,l_3)	
H	M	L	patterns for '1'
M	H	L	
H	H	L	
M	L	H	patterns for '0'
L	M	H	
L	L	H	
H	L	M	invalid patterns
L	H	M	
M	M	M	

Table 2. '1', '0' and invalid patterns (H: High, M: Middle, L: Low)

Our statistic results of the possible locations holding the specific frequencies are illustrated in Figure 3 as shadowed areas within a 8x8 block. In Table 1, we give our statistic results of the best location sets combined from these possible elements. The algorithms to write and read a label into and from an color or gray-scaled image are described in Algorithm 2(a)-(d).

Two parameters are provided for adjusting the robustness vs. modification visibility in an embedding process. The first one is the distance (D) between selected quantized frequency coefficients for representing an embedded bit. The default value of this distance is 1. The greater distance produces stronger robustness, but also may cause more serious modification visibilities. The second parameter is the quantization factor (Q) used to quantize the values selected for embedding code. The greater quantization factor results in less modifications to image data but weaker robustness against lossy JPEG compression. The default value of this quantization factor is 75%.

Algorithm 2(a): check_write(b,c_i)

(1) A three-element location set of the block *b* is pseudo-randomly (from the user key and image data) selected from the possible location sets listed in Table 1. They are denoted as (k_1,l_1), (k_2,l_2) and (k_3,l_3).

(2) The block *b* is locally DCT transformed and quantized at the locations (k_1,l_1), (k_2,l_2) and (k_3,l_3) with the quality factor Q parameter. Let $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$ and $Y_Q(k_3,l_3)$ be the quantized coefficient values at the selected locations.

(3) When c_i=1, if MIN ($|Y_Q(k_1,l_1)|$, $|Y_Q(k_2,l_2)|$) + MD < $|Y_Q(k_3,l_3)|$ where $|Y_Q(k_u,l_u)|$ is the absolute value of $Y_Q(k_u,l_u)$ with u ∈ 1..3, MIN is an operation that returns the minimum value of two elements, and MD is the maximum modification distance, then *b* is an invalid block:

(i) modify them to any of the invalid patterns shown in Table 2,

(ii) de-quantize and inversely transform (IDCT) them, and write them back to the block *b*,

(iii) return False.

(4) When c_i=0, if MAX ($|Y_Q(k_1,l_1)|$, $|Y_Q(k_2,l_2)|$) > $|Y_Q(k_3,l_3)|$ + MD where MAX is an operation that returns the maximum value of two elements, and MD is the maximum modification distance, then *b* is an invalid block:

(i) modify them to any of the invalid patterns shown in Table 2,

(ii) de-quantize, inversely transform (IDCT) them, and write them back to the block *b*,

(iii) return False.

(5) For other cases, return True.

Algorithm 2(b): check_read(b, c_i)

(1) A three-element location set of the block *b* is pseudo-randomly (from the user key and image data) selected from the possible location sets listed in Table 1. They are denoted as (k_1,l_1), (k_2,l_2) and (k_3,l_3).

(2) The block *b* is locally DCT transformed and quantized at the locations (k_1,l_1), (k_2,l_2) and (k_3,l_3) with the quality factor Q parameter. Let $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$ and $Y_Q(k_3,l_3)$ be the quantized coefficient values at the selected locations.

(3) If $|Y_Q(k_1,l_1)|$, $|Y_Q(k_2,l_2)|$ and $|Y_Q(k_3,l_3)|$ form any of the invalid patterns as illustrated in Table 2, return False, otherwise return True.

Algorithm 2(c): write(b, c_i)

Assume that a valid three-element location set of the block *b* has been pseudo-randomly selected. They are denoted as (k_1,l_1), (k_2,l_2), and (k_3,l_3). The block *b* is locally DCT transformed, and quantized at the locations (k_1,l_1), (k_2,l_2) and (k_3,l_3) with the quality factor Q. Let $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$, and $Y_Q(k_3,l_3)$ be the quantized coefficient values at the selected locations.

(1) When c_i=1, modify the $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$ and $Y_Q(k_3,l_3)$ such that they satisfy the following conditions: $Y_Q(k_1,l_1) > Y_Q(k_3,l_3) + D$, and $Y_Q(k_2,l_2) > Y_Q(k_3,l_3) + D$

(2) When c_i=0, modify the $Q(k_1,l_1)$, $Q(k_2,l_2)$ and $Y_Q(k_3,l_3)$ such that they satisfy the following conditions: $Y_Q(k_1,l_1) + D < Y_Q(k_3,l_3)$, and $Y_Q(k_2,l_2) + D < Y_Q(k_3,l_3)$

(3) $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$ and $Y_Q(k_3,l_3)$ are de-quantized, inversely transformed (IDCT), and written back to the block *b*.

Algorithm 2(d): read(b)

Assume that a valid three-element location set of the block *b* has been pseudo-randomly selected. They are denoted as (k_1,l_1), (k_2,l_2), and (k_3,l_3). The block *b* is locally DCT transformed, and quantized at the locations (k_1,l_1), (k_2,l_2) and (k_3,l_3) with the quality factor Q. Let $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$, and $Y_Q(k_3,l_3)$ be the quantized coefficient values at the selected locations.

(1) If $Y_Q(k_1,l_1) > Y_Q(k_2,l_2) + D$ and $Y_Q(k_2,l_2) > Y_Q(k_3,l_3) + D$, return 1.

(2) If $Y_Q(k_1,l_1) + D < Y_Q(k_3,l_3)$, and $Y_Q(k_2,l_2) + D < Y_Q(k_3,l_3)$, returns 0.

(3) In other cases, the embedded bit in this block *b* has been damaged.

3.2 Black/White Rate-Based Label Embedding for Binary Images

The value of each pixel in a binary image is either '1' or '0'. This determines that, in general, there is no 'noise' space which can be used for embedding additional information. To do it, we must find appropriate image areas where modifications for embedding labels do not affect seriously the quality of the original image. Obviously, these areas are varied with individual images or at least with types of images.

The proposed method for binary images is based on the ratio of '1' and '0' in a selected block. Suppose '1' represent black bit and '0' represent white bit in the source binary image. Let $P_1(b)$ be the rate (percentage) of blacks in the selected block *b*:

$$P_1(b) = \frac{N_1(b)}{64} \quad \text{where } N_1(b) \text{ is the number of '1' in the block } b.$$

Since the sum of percentages of blacks and whites in a block is 100%, the rate (percentage) of whites in the block *b* is $P_0(b) = 100 - P_1(b)$. A bit is embedded into a block *b* in the following way: a '1' is embedded into the block *b* if $P_1(b)$ is greater than a given threshold, and a '0' is embedded into the block *b* if $P_1(b)$ is less than another given threshold. A sequence of contiguous or distributed blocks is modified by switching whites to blacks or vice versa until such thresholds are reached.

We have classified two categories of binary images on which the generic method described above can be applied. These binary images are identified by distribution feature of blacks and whites. The first type of binary images is dithered image in which the black and white are well interlaced. The second type of binary images is black/white sharply contrasted images in which there exist clear boundaries between black and white areas.

Two modification strategies are adopted for these two types of binary images, respectively. For dithered binary images, modifications are well-distributed throughout the whole block: the bit that has most neighbors with the same value (either black or white) is reversed. For sharply contrasted binary images, modifications are carried out at the boundary of black and white pixels: the bit that has most neighbors with the opposite value is reversed. At the borders of the contiguous block, the neighbor bits in the neighbor blocks are also taken into account in both approaches. Two examples of both modification strategies are illustrated in Figure 4 and 5, respectively.

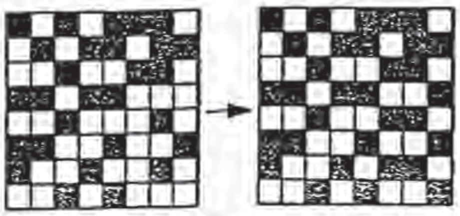


Figure 4. Well-distributed modifications

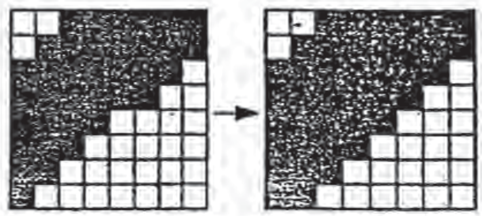


Figure 5. Modifications at black and white boundary

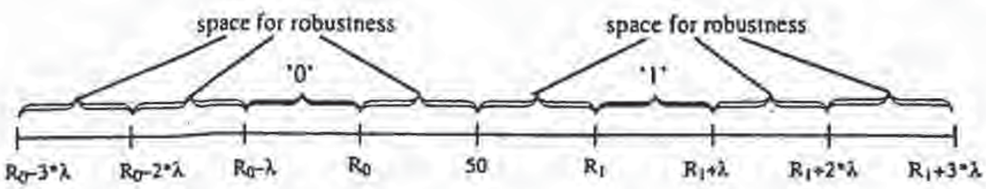


Figure 6. Achieve robustness in the black/white rate-based embedding method

Let R_1 be the threshold rate for '1'. Thus, the threshold rate for '0' is $R_0 = (100\% - R_1)$. Let λ be the robustness degree against image processing of labeled images. It represents the number of bits that can be altered after image processing without damage of embedded bits. For example, when λ is 5%, alternation (i.e. reversion from '1' to '0' or vice versa) of less than 4 bits in a block does not damage the embedded code. Our experiments have shown that the following values of them are the reasonable choices both in robustness of embedding code and the modification visibility:

$R_1 = 55$, $R_0 = 45$, and $\lambda = 5$.

The algorithms to write and read a label into and from a binary image are described in Algorithm 3(a)-(d).

Algorithm 3(a): check_write(b, c_i)

(1) If $P_1(b) > R_1 + 3*\lambda$ or $P_1(b) < R_0 - 3*\lambda$, return False.

(2) When c_i=1, if $P_1(b) < R_0$, modify the block *b* such that

$$P_1(b) < R_0 - 3*\lambda,$$

and then return False.

(3) When c_i=0, if $P_1(b) > R_1$, modify the block *b* such that

$$P_1(b) > R_1 + 3*\lambda,$$

and then return False.

(4) For other cases, return True.

Algorithm 3(b): check_read(b, c_i)

(1) If $P_1(b) > R_1 + 2*\lambda$ or $P_1(b) < R_0 - 2*\lambda$, return False.

(2) For other cases, return True.

Algorithm 3(c): write(b, c_i)

Assume that a valid block *b* has been pseudo-randomly selected. A bit c_i is embedded into *b* by switching blacks to whites or vise versa using the different modification strategies described above in order to reach a specified threshold rate.

(1) When c_i=1, modify the block *b* such that:

$$P_1(b) \geq R_1 \text{ and } P_1(b) \leq R_1 + \lambda.$$

(2) When c_i=0, modify the block *b* such that:

$$P_1(b) \leq R_0 \text{ and } P_1(b) \geq R_0 - \lambda.$$

(3) write the block *b* back to the image.

Algorithm 3(d): read(b)

Assume that a valid block *b* has been pseudo-randomly selected.

(1) If $P_1(b) > 50$, return 1.

(2) If $P_1(b) < 50$, return 0.

(3) For other cases, the embedded bit in the block *b* has been damaged.

4 Conclusion

The 'SysCoP' has been implemented on UNIX platform, and provides a graphical interface, a set of UNIX commands and an API (Application Programming Interface). It currently supports JPEG, PPM, GIF, and TIFF image formats. Experiments have been carried out to demonstrate the robustness of our methods against image processing. For the gray-scaled and color images using the JPEG-based embedding method, three images were labeled, and then processed by JPEG compression, format conversions and color reduction. In general, the results are quite satisfactory and meet the basic requirements for embedding codes as copyright labels. Due to the space limitation of the paper, concrete results are omitted. For the binary images, a dithered TIFF binary image and a sharply contrasted TIFF binary image were used in our tests. They are labeled first with R_1= 55% and the robustness degree (λ) 5%. The labeled TIFF images are then smoothed and converted to PBM. The embedded codes were not damaged in both labeled images after smoothing and conversions.

Our methods are still weak against physical damages (e.g. cut a pixel line, grab an area, etc.). Currently, we address this problem by allowing the user to specify 'valuable or sensitive' areas of an image into which labels are (repeatedly) embedded. Thus, cutting a part which is not in these areas does not damage embedded labels.

The methods described in this paper for embedding robust copyright labels for images have been extended to support MPEG-1. Two additional attacks in embedding copyright labels into MPEG-1 videos have been identified: removal of frames and re-compression with different patterns. To be resistant against them, the copyright label is repeatedly embedded into each frame. Thus we ensure that the label can be retrieved from each I-frame regardless of re-compression with different patterns. Furthermore, we are developing new labeling methods for other digital media, i.e. structured electronic documents (e.g. PostScript, SGML documents) and audio data. In addition, a WWW (World Wide Web) image copyright labeling server incorporating the methods described in this paper is being developed.

Acknowledge We are grateful to Scott Burgett from GMI, USA, who initiated and completed the JPEG-based embedding method of 'SysCoP' during his visiting stay at the Fraunhofer-IGD in Darmstadt. We also want to thank Martin Claviez and Joachim Krumb for helping us in implementing the 'SysCoP' system.

References

[1] J. BRASSIL, S. LOW, N. MAXEMCHUK, L. O'GORMAN. Electronic Marking and Identification Techniques to Discourage Document Copying. AT&T Bell Laboratories, Murray Hill, NJ, 1994.

[2] S. BURGETT, E. KOCH, J. ZHAO. A Novel Method for Copyright Labeling Digitized Image Data. Technical Report of Fraunhofer Institute for Computer Graphics, Darmstadt, August, 1994. (Also submitted to IEEE Trans. on Communication, September, 1994).

[3] A.K. CHOUDHURY, N.F. MAXEMCHUK, S. PAUL, H.G. SCHULZRINNE. Copyright Protection for Electronic Publishing over Computer Networks. AT&T Bell Laboratories, June 1994.

[4] W. DIFFIE and M. HELLMAN. New directions in cryptography. *IEEE Transactions on Information Theory*, vol. IT-22, pp. 644-654, 1976.

[5] R. C. DIXON. *Spread Spectrum Systems*. 2nd ed., Wiley, New York, NY, 1984.

[6] B. KAHIN. The strategic environment for protecting multimedia. *IMA Intellectual Property Project Proceedings*, vol. 1, no.1, 1994. pp.1-8.

[7] E. KOCH, J. RINDFREY, J. ZHAO. Copyright Protection for Multimedia Data. *Proceedings of the International Conference on Digital Media and Electronic Publishing* (6-8 December 1994, Leeds, UK).

[8] K. MANTUSI and K. TANAKA. Video-Steganography: How to secretly embed a signature in a picture. *IMA Intellectual Property Project Proceedings*, vol. 1, no. 1, 1994.

[9] W. NIBLACK, R. BARBER, W. EQUITZ, M. FLICKNER, E. GLASMAN, D. PETKOVIC, P. YANKER, C. FALOUTOS, G. TAUBIN. The QBIC Project: Querying Images By Content Using Color, Texture, and Shape. *SPIE* vol. 1908, 1993, pp.173-187.

[10] K.R. RAO and P. YIP. *Discrete Cosine Transform: Algorithms Advantages, Applications*. Academic Press. 1990.

[11] G.K. WALLACE. The JPEG still picture compression standard. *Communications of the ACM*, vol. 34, no. 4, April 1991. pp.30-40.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☑ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

4429 687

1995 IEEE WORKSHOP ON NONLINEAR SIGNAL AND IMAGE PROCESSING

Neos Marmaras
Greece
20 - 22 June 1995

Sponsored by

- Aristotle Univercity of Thessaloniki
- CERES Flying Dolphins
- Commission of the EU, Long Term Research
- Domaine Carras
- ESPRIT III Project NAT
- EURASIP
- Greek Ministry of Research and Technology
- IEEE CAS and ASSP Societies
- INTRACOM

ELECTRONIC WORKSHOP PROCEEDINGS

Papers listed by session

Invited papers

19.1. Towards Robust and Hidden Image Copyright Labeling

Session 19 : Image Signatures

Authors:E. Koch, J. Zhao

Fraunhofer Inst. for Computer Graphics

NSIP 95

HTML 2.0

(10)

Towards Robust and Hidden Image Copyright Labeling

E. Koch & J. Zhao
Fraunhofer Institute for Computer Graphics
Wilhelminenstr. 7, 64283 Darmstadt, Germany
email: {ekoch,zhao}@igd.fhg.de

Abstract — **This paper first presents the "hidden label" approach for identifying the ownership and distribution of multimedia information (image or video data) in digital networked environment. Then it discusses criteria and difficulties in implementing the approach. Finally a method using a JPEG model based, frequency hopped, randomly sequenced pulse position modulated code (RSPPMC) is described. This method supports robustness of embedded labels against several damaging possibilities such as lossy data compression, low pass filtering and/or color space conversion.**

1 Introduction

The electronic representation and transfer of digitized multimedia information (text, video, and audio) have increased the potential for misuse and theft of such information, and significantly increases the problems associated with enforcing copyrights on multimedia information [1,2]. These problems are rooted from the intrinsic features of the digitally formated information: (1) making copies is easy and inexpensive; (2) each copy is exactly identical to the original; and (3) distribution of the copies (e.g. via network or floppy) is easy and fast. For this reason, creators or publishers of multimedia materials fear providing their works for usage in new multimedia services, and are seeking technical solutions to the problems associated with copyright protection of multimedia data.

These problems have recently raised attentions in national IT (information technology) programmes, for example, NII (National Information Infrastructure) launched in the United States in 1993 established a working group on Intellectual Property Rights which is mainly concerned with copyright law and its application and effectiveness in the context of NII [3]. Several projects are currently or have recently been concerned with copyright and related issues in the digital world, for example, the EC ESPRIT project CITED (Copyright in transmitted electronic data) [4] and COPICAT (Copyright Ownership Protection in Computer Assisted Training) [5], and the EC RACE project ACCOPI (Access Control and Copyright Protection for Images) [6].

In [2], we have summarized that the technical mechanism of copyright protection for information in digital form can be divided into three levels: access control, use right control, and labeling-based mechanism. This paper addresses the problems in developing the last level of mechanism, and presents a JPEG-based method of labeling image for copyright protection.

Although some attention has been given to steganographic labeling and similar problems [7], there exists no technology designed to secretly embed a robust and invisible (hidden) copyright label in images. In particular, no current method adequately addresses the possibilities of using data compression, low pass filtering and/or simply changing the file format to remove an embedded code. Therefore, one of the main goals of this paper is to define a reasonable set of functional requirements and design criteria for an image copyright labeling method, and to furthermore demonstrate that the main difficulties involved in designing such a system can be solved.

The discussion begins with a section which outlines the functionality of the proposed system and general design criteria for the novel embedding technique. A specific method based on the JPEG compression standard for embedding copyright labels in image data is then presented.

2 Requirements and possible attacks

In order to be effective and workable in a multimedia environment, the copyright label must be difficult to remove and survive processing which does not seriously reduce the value of the image. This encompasses a wide range of possibilities including format conversions, data compression, and low pass filtering. In addition to copyright labeling of broadcast images, application areas for steganographic labeling techniques include copyright and/or secure records labeling of electronic publishing, facsimiles, scientific imaging, and medical imaging.

Requiring the copyright label to be a reliable property identification tool imposes following basic functional requirements on the system:

(1) The image must contain a label or code, which marks it as property of the copyright holder.

(2) The image data must contain a user code, which verifies the user is in legal possession of the data.

(3) The image data is labeled in a manner which allows its distribution to be tracked.

It is assumed, the main purpose of any attack would be to make the embedded label unverifiable. There are essentially two general ways to make the embedded label unverifiable: (1) alter the image data to render the copyright label unreadable, and (2) show that the label is not a reliable identification tool.

In addition several properties of digital data and design constraints, which are related to preventing attack on the copyright label, should be considered carefully. First, forgery of a digital copyright can only be prevented, if a forger cannot produce a valid copyright code. Second, the basic nature of digital images ensures that the copyright label can be easily altered if an attacker can identify the label data. Third, most digital images found in a multimedia environment can be low pass filtered, transformed to a different format or color space, or carefully re-quantized and compressed without significantly altering the images appearance or affecting its value. Finally, the image data is the only random sequence available to mask the data, and the statistics of the images, although generally unknown, are not under the control of the copyright system.

In sum, each of these points represents a potential means of attacking the copyright label and the following functional specifications are designed to prevent these attacks:

(1) A secret key type encryption code must be created using the unique identification of a work and used as the copyright label to prevent forgery of labeling.

(2) The image data must camouflage the copyright label code both visually and statistically to prevent an attacker from finding and deleting it. The functional requirement stating that the copyright label appears to be part of a normal image sequence and visually transparent is designed to prevent this attack.

(3) The signals used to embed the copyright label must contain a noise margin to resist damage if the image is processed or compressed.

(4) The system must be designed in such a way that the copyright labels locations and the same copyright code are not used repeatedly for embedding codes in different images to prevent the label from be found by comparing different images signed by the same owner.

The noise margin created by modeling the lossy compression allows for some loss of energy in the pulse, before the pulse becomes unreadable. Therefore if the pulse energy is concentrated at low frequencies, the embedded code should be relatively robust. Unfortunately, the final consideration with regard to pulse design and visual camouflaging, is in direct conflict with using low frequency pulse shapes. Specifically, it is widely accepted that noise in the low frequencies components of images is more noticeable than noise in the high frequency components. This is the basic concept behind the very efficient transform and sub-band coding techniques [8-10]. A reasonable trade-off between protection against processing attacks and visibility of the embedded code, is to make the pulses bandpass processes. Some additional design criteria must be developed to allow both requirements to be met simultaneously.

3 System Framework

The proposed approach, called Randomly Sequenced Pulse Position Modulated Code (RSPPMC) copyright labeling, is rooted in the well-known fact that typical digital images of people, buildings and natural settings can be considered as non-stationary statistical processes, which are highly redundant and tolerant of noise [8]. Hence, changes in the image data caused by moderate levels of wideband noise or controlled loss of information are hardly visibly noticeable, even when the altered images are compared directly with the original images.

Furthermore, the statistics of image sequences are only locally stationary and apriori unknown. More importantly, the process which produces such a sequence has random properties, which prevent the sequence from being reproduced exactly by a second experiment. This type of random signal is ideally suited for the purpose of statistically masking a sparse sequence of moderately large pulses.

The RSPPMC method consists of splitting the problem into two components. The first component produces the actual copyright code and a random sequence of locations for embedding the code in the image. This component is designed with the intention of implementing it, using existing encryption and pseudo random number generation techniques [11,12]. In fact,

BEST AVAILABLE COPY

these methods are only discussed to establish a framework for developing a novel technique for embedding data in images. The second component actually embeds the code at the specified locations, using a simple pulsing method, designed to appear to be a natural part of the image, which yet resists being damaged through simple processing techniques. This component consists of four steps:

(1) The position sequence is used to generate a sequence of pixel mapped locations where the code will be embedded.

(2) The blocks of 2-D image data, y(k,l) where k,l are the indices of discrete image points, are locally transformed and quantized at the locations selected in step 1, in a manner reflecting acceptable information loss in the image for the application to produce a 2-D image residual, n(k,l), in which the RSPPMC will actually be embedded.

(3) The code pulses, i.e. high or low, representing the binary code being embedded, are superimposed on the signal n(k,l) selected locations.

(4) The quantized data is decoded; and then, inversely transformed to produced the labeled image data.

In order to comply with functional requirements related to robustness, the transformation used in the second step includes the color space transformations and sub-banding and/or frequency transformations to allow direct access to the appropriate frequency bands in the gray scale component of an image. A quantization process is included in this step to guarantee that the label will survive a specific amount of information loss. A JPEG Compression Standard Based RSPPMC Copyright Label will be described in the next section.

4 Embedding a RSPPMC in Quantized JPEG Coefficients

Considering the functional requirement of robustness in a multimedia environment, the loss model in step 2 of the label process should be based on an industrial standard. From this perspective, image compression schemes used in GIF, TIFF, MPEG and JPEG are of interest. However, the wide spread use and growth of the JPEG [9] and MPEG formats and their efficiency in compressing images make transform coding the obvious choice for designing a copyright labeling system. Also, transform coding and/or sub-band coding techniques have the advantage of allowing direct access to specific frequency bands in the image, where the RSPPMC is to be embedded. This eliminates the problem of designing and detecting bandpass wavelets.

The basic characteristics of images, which make transform quantization a useful image data compression tool are (1) images are generally low pass processes, and (2) high frequency image components have little visual impact.

The DCT representation of images has been widely researched [10]. The typical characteristics of image DCT's are also well known. Readers unfamiliar with the DCT and image transform quantization should refer to [8-10] for details.

The second point allows the higher frequency coefficients to be more coarsely quantized than the low frequency components by the transform quantizer. Predictably, the JPEG transform quantizer utilizes this fact by increasing qs(k,l) as a function of the increasing frequency vector normal.

Using these assumptions, several signals can be derived from the image data Y(i,j), which naturally contain pulses meeting the requirements outlined in section 2. One of the simplest is the sub-block signal,

$$N(k_1,l_1,k_2,l_2) = |Y_Q(k_1,l_1)| - |Y_Q(k_2,l_2)| \quad (1)$$

where $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$ are the quantized coefficient values at the selected locations. This non-stationary random process should have an expected value of approximately zero if $|k_1,l_1|$ is approximately equal to $|k_2,l_2|$. Also, the signal should have a moderate variance level in the middle frequency ranges, i.e. $1.5 < |k,l| < 4.5$, where scattered changes in the image data should not be noticeably visible. The specific frequencies being used to embed the pulses will be "hopped" in this range to increase the robustness of the signal and making it more difficult to find. The principle being employed here is identical to the concept of frequency hopped spread spectrum communications [13].

A logical choice for the detection of "highs" and "lows", based on the signal defined in (1) is decided high if:

$$N(k_1,l_1,k_2,l_2) > 0, \quad (2a)$$

and decided low if,

$$N(k_1,l_1,k_2,l_2) < 0. \quad (2b)$$

However, embedding the code in this signal must also take into account the JPEG quantization process and any noise margin added to the pulses in the code. Therefore, the test for a written high is set as:

$$|Y_Q(k_1,l_1)| > |Y_Q(k_2,l_2)| + p. \quad (3a)$$

BEST AVAILABLE COPY

where p is a noise margin factor. The corresponding equation for n written low is:

$$|Y_Q(k_2,l_2)| > |Y_Q(k_1,l_1)| + p. \quad (3b)$$

Standard JPEG compression uses a "quality factor" to scale the quantization, allowing for different image qualities and compression factors. In order to guarantee that the copyright label will survive compressions up to a specific level compression, the quantization table should be scaled to the desired quality factor. Also, due to numerical problems (in calculating quantization step size according to quality factor, and in the quantization process) which can occur if the image is quantized with a JPEG quality greater than the designed factor for embedding the copyright code, some conditions must be be met. They are not discussed in this paper because of the limited space.

The method used to embed the copyright label in the sequence, $N(k_1,l_2,k_1,l_2)$, is not complicated. The high/low pulse pattern of the copyright label code is forced on the natural sequence at the selected group locations using a minimum mean square error approach, if it does not occur naturally. More complicated pulse pattern may be developed for representing the high/low bit, e.g. to use combinations (i.e. relationships) of three quantized elements $Y_Q(k_1,l_1)$, $Y_Q(k_2,l_2)$, $Y_Q(k_3,l_3)$ to replace equation (3).

In summary, the random pulse signal and conditions for detecting naturally occurring highs and lows described in equations (1) - (3) are designed to survive a JPEG compression down to a specified quality level. Clearly, decreasing the quality factor for the copyright code will make the signal more robust. However, this will also reduce the number of naturally occurring bits in the sequence. In addition, a lower quality factor will increase the likelihood that the changes necessary to superimpose the embedded code on the signal will be noticeably visible.

5 Conclusions

Using the prototypes we have developed, the experimental results indicate that the design requirements, developed in sections 2 for embedding a copyright label in image data, can be met, using the JPEG model based RSPPMC method developed in section 4. In particular, it was demonstrated that a copyright label code could be embedded in several images, using pulses with sufficient noise margins to survive common processing, such as lossy compression, color space conversion, and low pass filtering.

However, these results also indicate significant room for improvement in the method. One possibility for improvement could be to use different frequency band sets for encoding the high and low pulses. Also, methods could be developed to utilize image restoration techniques and pattern recognition techniques for verifying copyright labels. For example, pattern recognition techniques could be used to read copyright labels from images which have been cropped. In addition, methods suitable for applications with special requirements, such as cartography and medical imaging, are currently being investigated.

The authors would like to acknowledge Scott Burgett and Jochen Rindfrey, for their contributions to this work.

References

[1] B. Kahin, "The strategic environment for protecting multimedia", IMA Intellectual Property Project Proceedings, vol. 1, no.1, 1994, pp.1-8.

[2] E. Koch, J. Rindfrey, J. Zhao, "Copyright Protection for Multimedia Data", *Proceedings of the International Conference on Digital Media and Electronic Publishing* (6-8 December 1994, Leeds, UK).

[3] B.A. Lehman, R.H. Brown, "Intellectual Property and the National Information Infrastructure", Preliminary draft of the working group on intellectual property rights, July 1994

[4] G. Van Slype, Natural language version of the generic CITED model, ESPRIT II CITED Project 5469, June 28, 1994.

[5] A.J. Kitson and D.T. Seaton (eds.), Copyright Ownership Protection in Computer Assisted Training (COPICAT), Esprit Project 8195, Workpackage 2 (Requirements Analysis), Deliverable 1, June 2, 1994.

[6] RACE M 1005: Access control and copyright protection for images (ACCOPI), Workpackage 1 Deliverable, July 1994.

[7] K. Matsui and K. Tanaka, "Video-Steganography: How to secretly embed a signature in a picture," *IMA Intellectual Property Project Proceedings*, vol. 1, no. 1, 1994.

[8] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[9] G.K. Wallace, "The JPEG still picture compression standard", *Communications of the ACM*, vol. 34, no.4, April 1991, pp.30-40.

[10] K.R. Rao and P. Yip *Discrete Cosine Transform: Algorithms Advantages, Applications*, Academic Press, 1990.

[11] G. J. Simmons, *Contemporary Cryptology: The Science of Information Integrity*, IEEE Press, New York, 1994.

[12] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Son, Inc., New York et al., 1994.

[13] R. C. Dixon, *Spread Spectrum Systems*, 2nd ed., Wiley, New York, NY, 1984.

697

73

TOWARDS A ROBUST DIGITAL WATERMARK

R.G. van Schyndel(*), A.Z. Tirkel(+), C.F. Osborne(*)

(*) Department of Physics, Monash University, Clayton, 3168, Australia.
(+) Scientific Technology, P.O.Box 3018, E. Brighton, 3187, Australia.

Abstract

This paper discusses the feasibility of coding a robust, undetectable, digital water mark on a standard 512*512 intensity image with an 8 bit gray scale. The watermark is capable of carrying such information as authentication or authorisation codes, or a legend essential for image interpretation. This capability is envisaged to find application in image tagging, copyright enforcement, counterfeit protection, and controlled access. The method chosen is based on linear addition of the water mark to the image data. Originally, the authors made use of one dimensional encoding using m-sequences [1],[2], a process which, whilst showing promise, had considerable shortcomings. This paper presents an analysis of the one dimensional scheme and some constructions to extend this work to two dimensions.

1 Background

There exist two basic classes of electronic water marks: fragile and robust. Interest in both types has increased in recent times because of the explosion in digital communications and the rapidity and ease of transmission of electronic material which is subject to copyright. The authors have been concerned with the construction of the robust type, i.e. one which is resilient to some image distortions such as pixel or bit tampering, cropping, translation, rotation and shear. At this stage, our watermark possesses limited immunity against the first three distortions, but the intention is to improve its performance in the future. This should be contrasted with a novel technique involving a fragile watermark as described in [3], where, by deliberate design, *any* distortions render the watermark non-recoverable and this becomes proof of tampering. Both methods use LSB manipulation. Walton [3], also introduces an ingenious and effective palette manipulation technique to increase the watermark effectiveness by involving the complete RGB image components. A totally different technique and its variations is reviewed in [4]. Its major advantage is its compatibility with the JPEG format, whilst its principal disadvantage is that the watermark recovery requires the presence of the unencoded image. In this respect it differs from the other techniques. Our technique involves a linear addition of the watermark pattern, followed by a correlative recovery. Correlation can be defined as: cyclic or extended, global or character specific.

Correlation functions can be decomposed into: even and odd, or periodic and aperiodic. At this stage we are confined to binary characters only.

Our watermarks are chosen from two-dimensional array patterns based on m-sequences or extended m-sequences. An m-sequence basis is chosen because of their balance (zero mean), random appearance, resilience to filtering, cropping and individual bit errors, optimal autocorrelation properties and constrained cross-correlation. The water mark can be encoded by the choice of m-sequences and their phases.

2 Method

Our encoding method uses LSB addition for embedding the water mark [1], [2]. In many imaging systems the LSB is corrupted by hardware imperfections or quantisation noise and hence its manipulation is invisible, or of limited significance. The linear addition process is difficult to crack and makes it possible to embed multiple watermarks on the same image [2]. The decoding process makes use of the unique and optimal auto-correlation of m-sequence arrays to recover the watermark and suppress the image content. Since the correlation process involves averaging over long strings of binary digits, it is relatively immune to individual pixel errors, such as may occur in image transmission. The correlation process requires the examination of the complete bit pattern and must therefore be performed off-line, unless some form of dedicated, real-time, parallel processing is involved. The decoding process is not completely error free, due to partial correlation of the image data with the encoding sequence. In our previous work, we overcame this by filtering and dynamic range compression [2]. These artificial steps would be undesirable in a practical system. A typical 128*128 (unfiltered) image encoded with a one dimensional watermark is shown in Fig.1(Top left). The message is encoded on a line by line basis, using the ASCII character to select a sequence phase shift. There are numerous message repeats. The decoder output Fig.1(centre left) shows distinct message correlation peaks (white). Note that there are significant sidelobes due to image crosscorrelation effects. The top half of Fig.1. shows encoded images that have been progressively high-pass filtered, removing 10, 60 and 100 of the spatial frequency components from the total of 128. The watermark peaks survive all these filtering processes, demonstrating the robustness of the technique. The image content in the original and the decoded version is rendered negligible after the second or third of the filters. It is also clear that the filtering introduces progressively more severe ringing in the decoded output. This can result in ambiguities. We are presently investigating the feasibility of rectifying this shortcoming by the introduction of a matched filter in the decoding process. Fig.2. shows similar effects on the watermark alone (encoded on a null image). Contrast enhancement was employed to render the watermark visible.

3. Watermark requirements

An ideal watermark would possess:
(i) High in-phase autocorrelation peak for rows and columns [All]
(ii) Low out-of-phase autocorrelation for rows and columns [Costas Arrays]
(iii) Low cross-correlation between rows and between columns & between rows and columns [Perfect Maps, Hadamard Matrix, Legendre Arrays]
(iv) Low cross-correlation with image content [Folded M-sequence]
(v) Array diversity [Gold, Extended Gold Arrays]
(vi) Balance [All except Costas and some Gold]
The first two criteria are required for unambiguous watermark regisration, the third is necessary to avoid scrambling, the fourth minimises image related artefacts, whilst the fifth is concerned with the information capacity of the watermark. The sixth criterion maximises the significance of the correlation operation: in the binary case, the minority symbol determines the correlation score.
Constructions can be optimised for each of these requirements. However, a global optimisation requires compromise. We have examined all the criteria in detail, with the exception of (iv). Presently, we are examining methods of watermark design to minimise crosscorrelation with the image.

3.1 Image crosstalk suppression (ideal)

Clearly, it is possible to analyse the image content, by DCT or Walsh Transform and deduce a low crosscorrelation watermark by remapping any pattern, satisfying all criteria above except (iv). Similar effects could be assured by a random or adaptive search for a mapping to minimise the crosscorrelation with the image. However, such a procedure is impractical because there is no guarantee of uniqueness and hence the computation of the inverse mapping at the decoder.

3.2 Crosstalk suppression (practical)

There are at least three approaches which do not suffer from the above problem.

(1) Use longer m-sequences.
(2) Use high pass filtering.
(3) Use a "random" mapping.

The first is obvious. The other methods rely on the low overlap of the spatial frequency content of the image and watermark. In most cases (except random or fractal images), the image exhibits a (peaked) spatial frequency content constrained to low frequencies. By contrast, the m-sequence content is almost perfectly white. Therefore, as demonstrated by Fig.1., high pass filtering can reduce image related artefacts, without significantly degrading the peak.

3.3 Analysis

(3) requires an appreciation of the significant moments of the cross-correlation. Since the mean is subtracted from the image in the decoding stage, the correlation is that between two zero-mean functions and hence is itself zero. The variance, however is not so easily constrained. It is the main source of high cross-correlation peaks. Intuitively, this variance can be calculated by resolving each function into an orthogonal basis and applying random-phase statistics to each component. (A one dimensional analogy of this analysis is presented in the Appendix). The cross-correlation can therefore be expressed as a restricted summation over the m overlapping components. In the case of a "white" image, the total 2^n-1 components would contribute. Hence, assuming laws of large numbers, the ratio of variances is:

$$\frac{\sigma_w}{\sigma_{mw}} \approx \sqrt{\frac{m}{2^n - 1}} \qquad 1$$

A more complete analysis of these phenomena is presented in the Appendix.
For example, a linear image of 512 pixels can be expressed as a summation of 32 DCT components. The improvement offered by "whitening" is approximately a factor of 4.

It is not necessary to modify the image in order to implement this "whitening" process. It can just as easily be performed by embedding the m-sequence on the image with "random" pixel offsets, or by performing an orderly interleaving operation. The random offsets can be obtained from the m-sequence vector (n*1) for one dimensional or (n*m) for two dimensional patterns. We are presently investigating and comparing the performance of this technique against that of high-pass filtering.

4 Two-dimensional m-sequence based arrays

M-Sequences can be formed from starting vectors by a Fibonacci recursion relation. They are of maximal length (2^n-1) for a vector of length n. The autocorrelation function of an m-sequence is two valued: 2^n-1 (in phase), -1 (out of phase).

4.1 Extension of one dimensional arrays

A two-dimensional construction can be performed using a row by row phase shift. The effect on columns is that of decimation. Unique phase shifts as determined from Galois Field theory lead to the formation of columns, which are themselves m-sequences. The resulting array is an unbalanced Hadamard Matrix. Alternatively, a long sequence can be folded diagonally into an array format [5]. In this manner, the desirable one-dimensional autocorrelation property can be extended to two dimensions. The encoding and decoding performance of the Hadamard technique suffers from the image related effects because the correlations are performed on the (short and thus interference prone) row or column basis. The folded m-sequence is more immune to these effects, owing to its increased length. However, its information storage capacity is inferior. Watermarks encoded by both methods are presented, compared and analysed in the paper.

4.2 Intrinsic 2D constructions

We have also studied other fundamentally two-dimensional constructions. Costas Arrays are optimal in that their out-of-phase autocorrelation is minimum for shifts in either or both dimensions [6]. (Uniformly low sidelobe point- spread-function). They have been successfully deployed in radar and sonar, where time delays and frequency shifts (Doppler) can occur simultaneously. However, they are highly unbalanced and therefore prone to image related artefacts. Perfect Maps are constructions, where every m*n basis vector occurs once in a large pattern or map and hence can be used for automatic location. (An m-sequence is a one dimensional example of this category). The construction algorithm for Perfect Maps of large dimensions, commensurate with our image sizes is complicated [7]. However, some perfect maps are also Hadamard Matrices. We have examined examples of these, but still found them to be inadequate at rejecting image related artefacts. Legendre sequences and modified Legendre sequences, which are based on a quadratic residue (modulo n) and are similar to m-sequences of non-maximal length are also being studied. They are expected to improve on m-sequences for short lengths only. Extended m-sequences are attractive because they are commensurate with the image size (2^n). Whenever the extension by adding a zero to the m-sequence is performed to the longest run length of zeros, the resulting sequence still exhibits a strong in-phase autocorrelation peak of 2^n. This peak is surrounded by n zero values on either side, making it easy to recognise by filtering techniques. However, this is at the expense of numerous sidelobes at other phase shifts. The effect of these is being investigated. Gold Codes are linear additions of a preferred pair of m-sequences in with a prescribed relative phase shift. Alternatively, they can be viewed as sequences generated by a non-maximal feedback configuration shift register constructed to implement a product of the individual m-sequence generating polynomials. The family of codes generated by all the relative phase shifts and the original parent m-sequences in 2^n+1, of which approximately half are balanced. The auto and cross correlations are constrained to approximately $2^{n/2}$. These linear codes can be folded into array format, just as m-sequences. They offer greater information storage capacity because of their great diversity and constrained correlations. Gold codes can also be extended to length 2^n in a similar manner to m-sequences.

5 Conclusion

This paper presents a method of encoding and the recovery of a two-dimensional digital water mark on test images. The performance of the recovery process is analysed and improvements suggested.

6 References

[1] A.Z.Tirkel, G.A.Rankin, R.M.van Schyndel, W.J.Ho, N.R.A.Mee, C.F.Osborne. Electronic Water Mark. DICTA-93 Macquarie University, Sydney, December 1993. p.666-672.

[2] R.G. van Schyndel, A.Z.Tirkel, N.R.A.Mee, C.F.Osborne. A Digital Watermark. First IEEE Image Processing Conference, Houston TX, November 15-17, 1994, vol II, p.86-90.

[3] S.Walton. Image Authentication for a Slippery New Age. Dr.Dobb's Journal, April 1995, p.18-26, 82-87.

[4] F.M.Boland, J.K.K. Ó Rouanaidh and C.Dautzenberg. Watermarking Digital Images for Copyright Protection.

[5]. F.J. MacWilliams and N.J.A.Sloane. Pseudo-random Sequences and Arrays. Proc.IEEE, vol 64, 1715-1729, Dec.1976.

[6] S.W.Golomb and H.Taylor. Two-Dimensional Synchronization Patterns for Minimum Ambiguity. IEEE Trans. on Information Theory, vol IT-28, no.4, p.600-604, July 1982.

[7] T.Etzion. Construction for Perfect Maps and Pseudorandom Arrays. IEEE Trans. on Information Theory, vol 34, no 5. p.1308-1317. September 1988.

APPENDIX

Consider, for the sake of simplicity, a 512 pixel, one dimensional image I(x) and an encoding m-sequence M(x). This analysis can be extended to two dimensions. Both functions can be expressed as Fourier sums:

$$I(x) = \sum_{n=1}^{2^9-1} I_n \cos(\omega_n x + \phi_n) \quad \textbf{(A1)}$$

and since the m-sequence distribution is white,

$$M(x) = \sum_{n=1}^{2^9-1} m\cos(\omega_n x + \phi_n) \quad \textbf{(A2)}$$

The encoding process can be described by:

$$W(x) = I(x) - cM(x) \quad \textbf{(A3)}$$

where, in our case, c corresponds to LSB scaling

The decoding process involves the following:

1. Remove mean.
2. Perform correlation with a reference m-sequence in particular phase.
3. Repeat for all m-sequence phases.
4. Compute global correlation maximum and record m-sequence phase.

The correlation can be expressed as:

$$Y(\phi_m) = Y_m c + \sum_{i=0}^{2^9-1}\sum_{n=1}^{2^9-1} I_n m\cos(\omega_n x_i + \phi_n)$$

(A4)

where Y_m is the two-valued autocorrelation function of the m-sequence (2^9-1 for m=n, -1 otherwise). The first term contains information in n or the choice of sequence, whilst the summation term represents the interference (cross correlation with the image content). The value of m is determined by locating the maximum in $Y(\phi_m)$. In order to avoid false positive identification, missed detection and ambiguities, the image crosscorrelation peaks must be smaller than that of the m-sequence. This cross-correlation has zero mean, but its variance can be estimated as:

$$\sigma_Y = \langle Y(\phi_m) \rangle \quad \left[\sum_{i=0}^{2^9-1}\sum_{n=1}^{2^9-1} \frac{1}{2} I_n^2\right]^{\frac{1}{2}} \quad \textbf{(A5)}$$

Where the random phase approximation and the laws of large numbers have been implied. A value of $\sigma_Y < (2^9-1)/6$ will guarantee correct detection unconditionally for images with Gaussian statistics.

Most physical images contain only a few significant spectral components (L), around 0 frequency. Assuming a rectangular distribution,

$$\sigma_Y \approx \sigma_I \frac{2^{\frac{q}{2}-1}}{L^{\frac{1}{2}}} \qquad \text{(A6)}$$

where σ_I is the image variance, whose maximum is approximately 256/6 ≈ 43 for a 256 gray scale gaussian image.

There are two obvious methods of minimising σ_Y. High pass filtering the encoded image before performing the correlation will render the second term in (A4) negligible. However, the filter cut-in frequency is image-dependent. Also, this method introduces a degradation in the m-sequence autocorrelation peak (peak erosion) and an increase in sidelobe levels. There are also effects due to image content beyond the filter cut-in frequency (image leakage). Nevertheless, this method is capable of increasing the peak-to-sidelobe ratio by a factor of 2.

The second method of relative "whitening" described in the main text does not suffer from the above deficiencies, but does affect the spatial properties of the m-sequence. It is not clear if both techniques can be used in cascade, nor if they are commutative.

686

(14)

A TWO-DIMENSIONAL DIGITAL WATERMARK

A.Z.Tirkel(+), R.G.van Schyndel(*), C.F.Osborne(*)

(+)Scientific Technology,
P.O.Box 3018, Dendy Brighton, 3186, Australia.

(*) Department of Physics, Monash University,
Clayton, 3168, Australia.

ABSTRACT

This paper discusses the feasibility of coding a robust, undetectable, digital water mark on a standard 512*512 intensity image with an 8 bit gray scale. The watermark is capable of carrying such information as authentication or authorisation codes, or a legend essential for image interpretation. This capability is envisaged to find application in image tagging, copyright enforcement, counterfeit protection, and controlled access. The method chosen is based on linear addition of the water mark to the image data. Originally, the authors made use of one dimensional encoding using m-sequences [1],[2], a process which, whilst showing promise, had considerable shortcomings. This paper analyses constructions to extend this work to two dimensions and discusses compatibility of the technique with JPEG image transmission.

1 Review

There exist two basic classes of electronic water marks: fragile and robust. Interest in both types has increased in recent times because of the explosion in digital communications and the rapidity and ease of transmission of electronic material which is subject to copyright. The authors have been concerned with the construction of the robust type, i.e. one which is resilient to some image distortions such as pixel or bit tampering, cropping, translation, rotation and shear. Another form of robustness concerns lossy compression processing, such as coding via the Discrete Cosine Transform, such as JPEG. At this stage, our watermark possesses limited immunity against the first three pixel related distortions. In this paper, we discuss methods of addressing JPEG compatibility.

Our watermarking method differs significantly from the novel technique recently introduced by Walton [3]. This involves a fragile watermark, where, by deliberate design, *any* distortions render the watermark non-recoverable and this becomes proof of tampering. Both methods use LSB manipulation. Walton [3], also introduces an ingenious and effective palette manipulation technique to increase the watermark effectiveness by involving the complete RGB image components.

A totally different technique and its variations is reviewed in [4]. Its major advantage is its compatibility with the JPEG format, whilst its principal disadvantage is that the watermark recovery requires the presence of the unencoded image. In this respect it differs from the other techniques.

Our technique involves a linear addition of the watermark pattern, followed by a correlative recovery. Correlation can be defined as: cyclic or extended, global or character specific.

Correlation functions can be decomposed into: even and odd, or periodic and aperiodic. At this stage we are confined to binary characters only. Our watermarks are chosen from two-dimensional array patterns based on m-sequences or extended m-sequences. An m-sequence basis is chosen because of their balance (zero mean), random appearance, resilience to filtering, cropping and individual bit errors, optimal autocorrelation properties and constrained cross-correlation. The water mark can be encoded by the choice of m-sequences and their phases.

2 Method

Our encoding method uses LSB addition for embedding the water mark [1], [2]. In many imaging systems the LSB is corrupted by hardware imperfections or quantisation noise and hence its manipulation is invisible, or of limited significance. The linear addition process is difficult to crack and makes it possible to embed multiple watermarks on the same image [2]. The decoding process makes use of the unique and optimal auto-correlation of m-sequence arrays to recover the watermark and suppress the image content. Since the correlation process involves averaging over long strings of binary digits, it is relatively immune to individual pixel errors, such as may occur in image transmission. The correlation process requires the examination of the complete bit pattern and must therefore be performed off-line, unless some form of dedicated, real-time, parallel processing is involved. The decoding process is not completely error free, due to partial correlation of the image data with the encoding sequence. In our previous work, we overcame this by filtering and dynamic range compression [2]. These artificial steps would be undesirable in a practical system. A typical 128*128 (unfiltered) image encoded with a one dimensional watermark is shown in Fig.1(Top left). The message is encoded on a line by line basis, using the ASCII character to select a sequence phase shift. There are numerous message repeats. The decoder output Fig.1(centre left) shows distinct message correlation peaks (white). Note that there are significant sidelobes due to image crosscorrelation effects. The top half of Fig.1. shows encoded images that have been progressively high-pass filtered, removing 10, 60 and 100 of the spatial frequency components from the total of 128. The watermark peaks survive all these filtering processes, demonstrating the robustness of the technique. The image content in the original and the decoded version is rendered negligible after the second or third of the filters. It is also clear that the filtering introduces progressively more severe ringing in the decoded output. This can result in ambiguities. We are presently investigating the feasibility of rectifying this shortcoming by the introduction of a matched filter in the decoding process.

3. Watermark properties

An ideal watermark would possess:

(i) High in-phase autocorrelation peak for rows and columns [All]
(ii) Low out-of-phase autocorrelation for rows and columns [Costas Arrays]
(iii) Low cross-correlation between rows and between columns & between rows and columns [Perfect Maps, Hadamard Matrix, Legendre Arrays]
(iv) Low cross-correlation with image content [Folded M-sequence]
(v) Array diversity [Gold, Extended Gold Arrays]
(vi) Balance [All except Costas and some Gold]
(vii) Compatibility with standard image transmission format such as JPEG [Folded M-sequence].
(viii) Long span in order to prevent unauthorised cracking. [GMW codes]

The first two criteria are required for unambiguous watermark registration, the third is necessary to avoid scrambling, the fourth minimises image related artefacts, whilst the fifth is concerned with the information capacity of the watermark. The sixth criterion maximises the significance of the correlation operation: in the binary case, the minority symbol determines the correlation score.
The seventh criterion requires robustness against the low-pass filtering along a diagonal raster. It is described in more detail in 3.2. The eighth criterion relates to code inversion property. All codes can be generated by a recursion relation and this can be deduced from a sample of the code by solution of a set of simultaneous equations (matrix inversion). The minimum number of terms required for unambiguous inversion is called the span. M-sequences have a short span of 2n, where n is the order of the polynomial describing the recursion relation. This is because of their linear nature. GMW codes

use non-linear recursion, which is optimised to yield much larger spans, with minimum impact on sequence properties. They are therefore ideal in situations where security is paramount.
Constructions can be optimised for each of these requirements. However, a global optimisation requires compromise. We have examined all the criteria in detail with particular reference to (iv) [8] and (vii).

3.1 Image crosstalk suppression

Clearly, it is possible to analyse the image content, by DCT or Walsh Transform and deduce a low crosscorrelation watermark by remapping any pattern, satisfying all criteria above except (iv). Similar effects could be assured by a random or adaptive search for a mapping to minimise the crosscorrelation with the image. However, such a procedure is impractical because there is no guarantee of uniqueness and hence the computation of the inverse mapping at the decoder.
There are at least three approaches which do not suffer from the above problem.

(1) Use longer m-sequences.
(2) Use high pass filtering.
(3) Use a "random" mapping.

The first is obvious (longer averagimg). The other methods rely on the low overlap of the spatial frequency content of the image and watermark. In most cases (except random or fractal images), the image exhibits a (peaked) spatial frequency content constrained to low frequencies. By contrast, the m-sequence content is almost perfectly white, as shown in Fig.2. Therefore, as demonstrated by Fig.1., high pass filtering can reduce image related artefacts, without significantly degrading the peak.

3.2 JPEG compatibility

As already demonstrated in Fig.1, the watermark is resilient against severe (0.25 quality factor) DCT high-pass filtering. Since the watermark mask is almost perfectly (spectrally) white, the same is true about low-pass filtering. In order to preserve this feature in raster format, the m-sequence should be embedded in a commensurate diagonal manner. The folded m-sequence of [5] is ideal for this purpose. The partitioning of the process into 8*8 blocks should pose no significant problems. The m-sequence employed in Fig.1. was 4 times longer than the linear dimension of the image, with no discernible effects on the result. We have actually experimented with 8*8 blocks and found no surprises. The only disadvantage of JPEG processing is that the high-pass filtering method of image-related artefacts (section 3.1) is incompatible, with the low-pass filtering involved in image compression. The same is likely to apply to the random mapping technique. Hence, the suppression of image related effects can only be achieved by the use of longer sequences. This imposes a limit on the information content of the watermark. The effects of sequence length on information content have been discussed in [1].
Another feature of JPEG processing is the capability of performing image manipulations on-line. This poses no problems at the encoding stage of our watermark. However, the watermark recovery process requires the execution of a sliding correlation to determine the location of a global maximum. At present, this process is being performed sequentially and hence off-line. We are investigating hardware and software techniques to render these operation parallel. Alternatively, a DCT-based correlation computation could be devised. This is also being examined.

4 Two-dimensional m-sequence based arrays

M-Sequences can be formed from starting vectors by a Fibonacci recursion relation. They are of maximal length (2^n-1) for a vector of length n. The autocorrelation function of an m-sequence is two valued: 2^n-1 (in phase), -1 (out of phase).

4.1 Extension of one dimensional arrays

A two-dimensional construction can be performed using a row by row phase shift. The effect on columns is that of decimation. Unique phase shifts as determined from Galois Field theory lead to the formation of columns, which are themselves m-sequences. The resulting array is an unbalanced Hadamard Matrix. Alternatively, a long sequence can be folded diagonally into an array format [5]. In this manner, the desirable one-dimensional autocorrelation property can be extended to two dimensions. The encoding and decoding performance of the Hadamard technique suffers from the image related effects because the correlations are performed on the (short and thus interference prone) row or column basis. The folded m-sequence is more immune to these effects, owing to its increased length. However, its information storage capacity is inferior. Watermarks encoded by both methods are presented, compared and analysed in the paper.

4.2 Intrinsic 2D constructions

We have also studied other fundamentally two-dimensional constructions. Costas Arrays are optimal in that their out-of-phase autocorrelation is minimum for shifts in either or both dimensions [6]. (Uniformly low sidelobe point- spread-function). They have been successfully deployed in radar and sonar, where time delays and frequency shifts (Doppler) can occur simultaneously. However, they are highly unbalanced and therefore prone to image related artefacts. Perfect Maps are constructions, where every m*n basis vector occurs once in a large pattern or map and hence can be used for automatic location. (An m-sequence is a one dimensional example of this category). The construction algorithm for Perfect Maps of large dimensions, commensurate with our image sizes is complicated [7]. However, some perfect maps are also Hadamard Matrices. We have examined examples of these, but still found them to be inadequate at rejecting image related artefacts. Legendre sequences and modified Legendre sequences, which are based on a quadratic residue (modulo n) and are similar to m-sequences of non-maximal length are also being studied. They are expected to improve on m-sequences for short lengths only. Extended m-sequences are attractive because they are commensurate with the image size (2^n). Whenever the extension by adding a zero to the m-sequence is performed to the longest run length of zeros, the resulting sequence still exhibits a strong in-phase autocorrelation peak of 2^n. This peak is surrounded by n zero values on either side, making it easy to recognise by filtering techniques. However, this is at the expense of numerous sidelobes at other phase shifts. The effect of these is being investigated. Gold Codes are linear additions of a preferred pair of m-sequences with a prescribed relative phase shift. Alternatively, they can be viewed as sequences generated by a non-maximal feedback configuration shift register constructed to implement a product of the individual m-sequence generating polynomials. The family of codes can be generated by all the relative phase shifts and the original parent m-sequences in 2^n+1, of which approximately half are balanced. The auto and cross correlations are constrained to approximately $2^{n/2}$. These linear codes can be folded into array format, just as m-sequences. They offer greater information storage capacity because of their great diversity and constrained correlations. Gold codes can also be extended to length 2^n in a similar manner to m-sequences.

4.3 Extensions to Multi-Dimensional Arrays

So far, our watermarking scheme has been confined to one and two-dimensional spatial constructions employing a gray scale image. Extensions to colour (RGB) encoding have the potential of enlarging the dimensionality to a total of 5. This could be employed for:

(i) Increasing the information content of the watermark. For example, three independent, two-dimensional messages could be encoded instead of one.
(ii) Increasing the length of the watermark code to reduce image related effects.
(iii) Redundancy coding.

(iv) Novel, multi-dimensional array coding.
(v) Non-binary character sequences.

These aspects are presently being evaluated.

4.4 Non-imaging applications

The watermarking technique discussed here has potential applications to audio copyright protection and audio system and equalisation control. Two one-dimensional patterns can be embedded in each of the stereo channels on CD-ROM or DAT. These codes could be designed to have a deliberately long span (such as GMW codes), in order to prevent cracking. This technique offers potential resistance to resampling/subsampling, which are akin to scaling/rotation. These codes could also be employed in automatic spectral and delay calibration/equalisation of the sound system, because of their optimal impulse response. This feature could be particularly useful in dynamic situations, where the audio environment is constantly changing.

5 Conclusion

This paper presents a method of encoding and recovery of a two-dimensional digital water mark on test images. The compatibility of the watermarking process with JPEG coding is discussed.

6 Acknowledgements

The authors would like to extend their gratitude to Nicholas Mee and Gerard Rankin for their assistance in mathematical theory and computer based sequence generation and analysis respectively. Their contributions have been invaluable.

7 References

[1] A.Z.Tirkel, G.A.Rankin, R.M.van Schyndel, W.J.Ho, N.R.A.Mee, C.F.Osborne. Electronic Water Mark. DICTA-93 Macquarie University, Sydney, December 1993. p.666-672.
[2] R.G. van Schyndel, A.Z.Tirkel, N.R.A.Mee, C.F.Osborne. A Digital Watermark. First IEEE Image Processing Conference, Houston TX, November 15-17, 1994, vol II, p.86-90.
[3] S.Walton. Image Authentication for a Slippery New Age. Dr.Dobb's Journal, April 1995. p.18-26, 82-87.
[4] F.M.Boland, J.K.K. Ó Rouanaidh and C.Dautzenberg. Watermarking Digital Images for Copyright Protection.
[5]. F.J. MacWilliams and N.J.A.Sloane. Pseudo-random Sequences and Arrays. Proc.IEEE, vol 64, 1715-1729, Dec.1976.
[6] S.W.Golomb and H.Taylor. Two-Dimensional Synchronization Patterns for Minimum Ambiguity. IEEE Trans. on Information Theory, vol IT-28, no.4, p.600-604, July 1982.
[7] T.Etzion. Construction for Perfect Maps and Pseudorandom Arrays. IEEE Trans. on Information Theory, vol 34, no 5, p.1308-1317. September 1988.

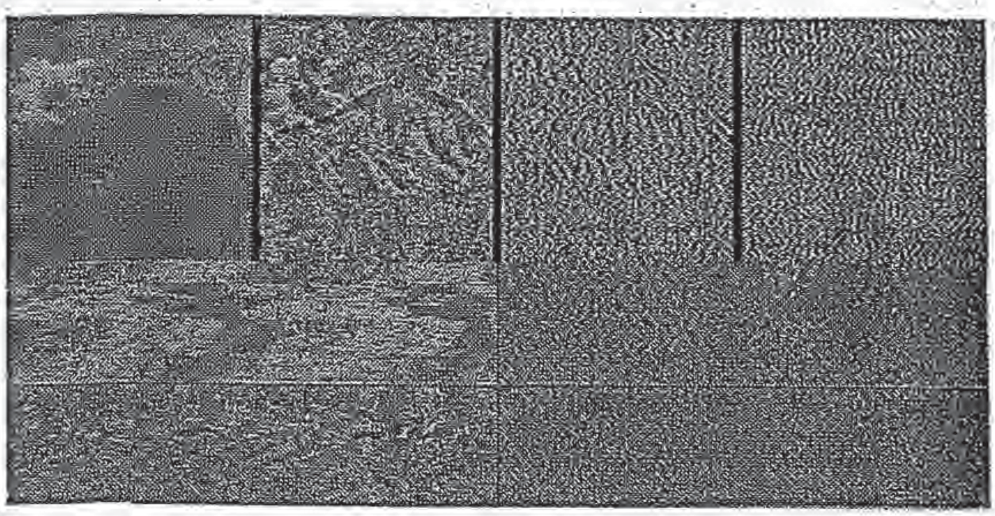


Figure 1
Upper (left to right: Encoded image after high pass filtering, removing
(a) 0, (b) 10, (c) 60, (d) 100 of 128 Spatial Frequency Components
Lower (Centre Left, Bottom Left, Centre Right, Bottom Right) : Corresponding Decoded Patterns
(Medium gray=0, darker=negative, lighter=positive - all image intensities have been suitably scaled)

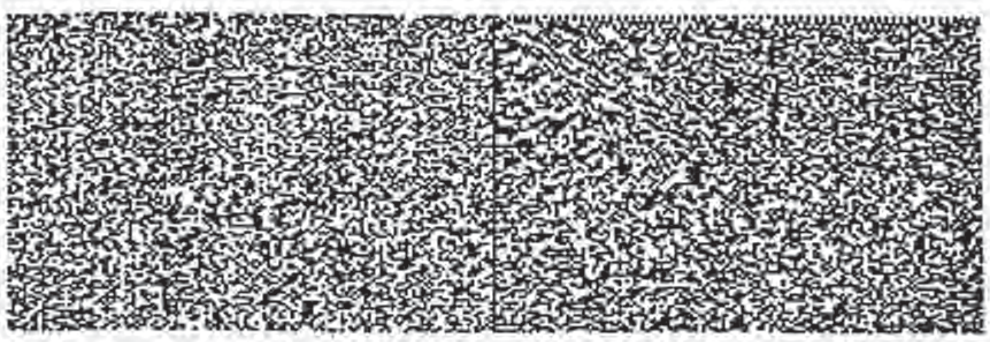


Fig 2.
Left - DCT of watermark
Right - DCT of image
(Interpretation as in Fig 1)

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☑ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

(15)

IMAGE WATERMARKING - A SPREAD SPECTRUM APPLICATION

Anatol.Z.Tirkel* (Senior Member), Charles F Osborne, Ron G. van Schyndel.

*Scientific Technology, 3/9 Barnato Gve., Armadale 3143 Australia
Department of Physics, Monash University, Clayton 3168 Australia
Andrew.Tirkel@sci.monash.edu.au

ABSTRACT

This paper discusses the feasibility of coding a robust, undetectable, digital water mark on a standard 512*512 intensity image with an 24 bit RGB format. The watermark is capable of carrying such information as authentication or authorisation codes, or a legend essential for image interpretation. This capability is envisaged to find application in image tagging, copyright enforcement, counterfeit protection, and controlled access. The method chosen is based on linear addition of the water mark to the image data. The patterns adopted to carry the watermark are adaptations of m-sequences in one and two dimensions. The recovery process is based on correlation, just as in standard spread spectrum receivers. The technique is quite successful for one dimensional encoding with binary patterns, as shown for a variety of gray scale test images. A discussion of extensions of the method to two dimensions, RGB format and non-binary alphabets is presented. A critical review of other watermarking techniques is included.

1 BACKGROUND

The art of hiding messages in written text was known to the ancient Greeks as steganography. Many ingenious schemes to achieve that objective have been devised over the centuries. However, the more recent development of computer technology and the proliferation of image and graphics type data have generated the capability and the motivation for electronic watermarking as a means of copyright protection. There exist two basic classes of electronic water marks: fragile and robust. This paper is concerned with the construction of the robust type, i.e. one which is resilient to some image distortions such as pixel or bit tampering, cropping, translation, rotation and shear. At this stage, such a watermark possesses limited immunity against the first three distortions, but the intention is to improve its performance in the future. This should be contrasted with a novel technique involving a fragile watermark as described in [3], where, by deliberate design, *any* distortions render the watermark non-recoverable and this becomes proof of tampering. Both methods use LSB manipulation. Walton [3], also introduces an ingenious and effective palette manipulation technique to increase the watermark effectiveness by involving the complete RGB image components. A totally different technique and its compatibility with the JPEG format, whilst its principal disadvantage is that the watermark recovery requires the presence of the unencoded image. In this respect it differs from the other techniques. Other techniques being investigated are concerned with encryption of JPEG bit stream and involve the use of checksums [14]. The technique described here involves a linear addition of the watermark pattern, followed by a correlative recovery. Correlation can be performed as cyclic or extended, global or character specific operations. Novel methods of defining correlation can be devised. The traditional decomposition of correlation functions into even and odd, or periodic and aperiodic components does not apply, because the embedding pattern has periodicity commensurate with that of the image. So far, the watermarks have been chosen from one and two-dimensional array patterns based on m-sequences or extended m-sequences [5]. An m-sequence basis is chosen because of their balance (zero mean), random appearance, optimal autocorrelation properties and constrained cross-correlation. The water mark has been encoded by the choice of m-sequences and their phases.

2 METHOD

The encoding method uses LSB addition for embedding the water mark [1], [2], [8], [13]. A similar method has since been developed commercially by Digimarc [10], who add random multiples of the LSB on a pixel by pixel basis. Their decoding process is subtractive in the presence of the unencoded image and seems to be correlative in its absence. The extension of the scheme described here to multiple LSB's has been considered to be an integral part of the transition to a non-binary alphabet, such as that offered by the RGB format. The restriction to LSB manipulation has certain advantages, since in many imaging systems the LSB is corrupted by hardware imperfections or quantisation noise and hence this form of implementation renders the watermark invisible. Our present technique involves the addition of unfiltered m-sequences, although it is possible to devise a matched filter, such that the spectral components of the watermark match those of the image. This reduces the visibility of the watermark and permits the use of higher order bits in the encoding process. Another benefit of such filtering is that it ensures that any distortions due to lossy image compression or transmission errors affect the watermark and the image equally. Therefore, as long as the image is acceptable, so is the watermark.

The only significant case where watermarks are readily detectable is that of computer generated images, which are free of noise. In that instance, other means of

BEST AVAILABLE COPY

watermarks on the same image [2]. The decoding process makes use of the unique and optimal auto-correlation of m-sequence arrays to recover the watermark and suppress the image content. Since the correlation process involves averaging over long strings of binary digits, it is relatively immune to individual pixel errors, such as may occur in image transmission. The correlation process requires the examination of the complete bit pattern and must therefore be performed off-line, unless some form of dedicated, real-time, parallel processing is involved. Presently, two image processing hardware platforms (SGS Thomson IMSA100 and a Philips OPTIC-Optimized Pixel Template Image Correlator) are being evaluated as candidates for on-line performance of the decoding algorithm. The decoding process is not completely error free, due to partial correlation of the image data with the encoding sequence. The presence of significant correlation betweeen the image and the watermark typically results in false peaks and true peak erosion. This in turn can result in ambiguous or false decoding. In previous work, this was overcome by filtering and dynamic range compression [2]. These artificial steps would be undesirable in a practical system. Other means such as redundancy coding restrict the data content of the watermark. Morphological methods of peak detection, based on the unique neighbourhood characteristics of the pixel corresponding to the peak correlation have been evaluated, but so far the improvement attributable to them appears similar to that of filtering. Neural nets trained for local peak detection are another option for future evaluation.

A typical 128*128 (unfiltered) image encoded with a one dimensional watermark is shown in Fig.1(Top left). The message is encoded on a line by line basis, using the ASCII character to select a sequence phase shift. There are numerous message repeats. The decoder output Fig.1(centre left) shows distinct message correlation peaks (white). Note that there are significant sidelobes due to image crosscorrelation effects. The top half of Fig.1, shows encoded images that have been progressively high-pass filtered, removing 10, 60 and 100 of the spatial frequency components from the total of 128. The watermark peaks survive all these filtering processes, demonstrating the robustness of the technique. The image content in the original and the decoded version is rendered negligible after the second or third of the filters. A different presentation of this process is shown in Fig.2.

3. WATERMARK PROPERTIES

An ideal watermark would possess:

(i) High in-phase autocorrelation peak for rows and columns
(ii) Low out-of-phase autocorrelation for rows and columns.
(iii) Low cross-correlation between rows and between columns & between rows and columns
(iv) Low cross-correlation with image content
(v) Array diversity
(vi) Balance
(vii) Compatibility with standard image transmission format such as JPEG
(viii) Long span, in order to prevent unauthorised cracking

The first two criteria are required for unambiguous ...

whilst the fifth is concerned with the information capacity of the watermark. The sixth criterion maximises the significance of the correlation operation: in the binary case, the minority symbol determines the correlation score.

The seventh criterion requires robustness against the low-pass filtering along a diagonal raster. The eighth criterion relates to code inversion property. All codes can be generated by a recursion relation and this can be deduced from a sample of the of the code by solution of a set of simultaneous equations (matrix inversion). The minimum number of terms required for unambiguous inversion is called the span. M-sequences have a short span of 2n, where n is the order of the polynomial describing the recursion relation. This is because of their linear nature. GMW codes use non-linear recursion, which is optimised to yield much larger spans, with minimum impact on sequence properties. They are therefore ideal in situations where security is paramount. The sidelobe performance of these has not yet been evaluated. A search for a mapping to convert two dimensional arrays into GMW format is continuing.

Constructions can be optimised for each of these requirements. However, a global optimisation requires compromise. All criteria have been examined in detail with particular reference to (iv) [8] and (vii).

4 TWO-DIMENSIONAL M-SEQUENCE ARRAYS

M-Sequences can be formed from starting vectors by a Fibonacci recursion relation. They are of maximal length i.e. (2^n-1) for a vector of length n. Typically, the alphabet of symbols used to generate the sequence forms a finite base field, a Galois Field (GF). In most applications, binary or binary derived base fields such as GF(2) are involved. A good review of non-binary bas field applications can be found in [12]. The recursion relation can be described by a generating polynomial over the GF. These polynomials, whose roots are not elements of the base field, themselves form an extension field. Their solutions (in the extension field) are powers of each other, which is equivalent to sequences being decimations of each other.

Two dimensional patterns are generated by polynomials in two variables. This is equivalent to a two-dimensional shift register. One dimensional polynomials have been studied extensively, whilst higher dimensional constructions have been devised ad-hoc, with specific applications in mind. [5] is one of the few references which attempts to treat this problem and its extensions to base fields other than GF(2).

4.1 SOME TWO-DIMENSIONAL CONSTRUCTIONS

The autocorrelation function of binary m-sequence is two valued: 2^n-1 (in phase), -1 (out of phase). A two-dimensional construction can be performed using a row by row phase shift. The effect on columns is that of decimation. Unique phase shifts as determined from Galois Field theory lead to the formation of columns, which are themselves m-sequences. The resulting array is an unbalanced Hadamard Matrix. Alternatively, a long sequence can be folded diagonally into an array format [5].

In this manner, the desirable one-dimensional autocorrelation property can be extended to two the Hadamard technique suffers from the image related effects because the correlations are performed on the (short and thus interference prone) row or column basis. The folded m-sequence is more immune to these effects, owing to its increased length. However, its information storage capacity is inferior. We have encoded watermarks by both methods and have found them lacking.

There exist other fundamentally two-dimensional constructions. Costas Arrays are optimal in that their out-of-phase autocorrelation is minimum for shifts in either or both dimensions [6]. (Uniformly low sidelobe point-spread-function). They have been successfully deployed in radar and sonar, where time delays and frequency shifts (Doppler) can occur simultaneously. However, they are highly unbalanced and therefore prone to image related artefacts. Perfect Maps are constructions, where every m*n basis vector occurs once in a large pattern or map and hence can be used for automatic location. (An m-sequence is a one dimensional example of this category). The construction algorithm for Perfect Maps of large dimensions, commensurate with our image sizes is complicated. However, some perfect maps are also Hadamard Matrices. We have examined examples of these, but still found them to be inferior at rejecting image related artefacts.

4.2 EXTENSIONS TO NON-BINARY ALPHABETS

The watermarking scheme demonstrated in the diagrams has been confined to one and two-dimensional spatial constructions employing a gray scale image. Extensions to colour (RGB) encoding have the potential of expanding the capabilities. This could be employed for:

(I) Increasing the information content of the watermark. For example, three independent, two-dimensional messages could be encoded instead of one.
(ii) Increasing the length of the watermark code to reduce image related effects.
(iii) Redundancy coding.
(iv) Non-binary character sequences.

The last feature is of particular interest because of the difference between the encoding process of the watermark and the standard embedding of the spreading code on the carrier as practiced in spread spectrum communications. There, the use of QPSK calls for GF(4) as a natural base field. It also permits the use of an isomorphism of the characters with complex roots of unity to derive convenient constructions of the complex correlation. The existence of two quadrature carriers is beneficial, but is quite irrelevant to the watermarking scheme. The RGB format permits the use of GF(8) as a base field. An example of a two dimensional RGB pattern based on GF(8) is shown in the presentation. This example uses the multiplication table based on each character being associated with a power of a primitive root of unity. This is not an essential requisite. A counter example is demonstrated by [12], who considers all possible algebras over GF(4) for communications applications. Many of these algebras are not based on roots of unity. In our future work, we propose to examine GF(8) in a similar manner. It may even be practical to combine two LSB's of RGB channels to construct a character set based on Galois Fields of dimension 64. These character sets can be generated from GF(2) by numerous field degeneracies to construct arrays which suppress the undesirable two-dimensional symmetries present in the McWilliams and Sloane folded m-sequence construction. These unattractive symmetries and a leading blank row/column are shown to be present in GF(8) and survive transformation mappings based on m-sequences, as is shown in the presentation. Green [15] devises constructions of large size over GF(2), which avoid these problems, but the minimum array size (91*45) and its aspect ratio is not conducive to imaging applications. It may be possible to construct square arrays over GF(8) or GF(64) based on algebraic degeneracy or adapt perfect maps to those non-binary character sets. In fact, it may be possible to devise a distance based correlation measure, as opposed to the use of complex multiplication. The merits of such a technique are still being investigated. It may also be feasible to reduce the spectral occupancy of the watermark by modulating RGB components alternately and using differential coding, as in a more generalised form of OQPSK or Frank coding. Such a scheme could be incorporated into the JPEG conversion table. There may be applications of such techniques to spread spectrum communications, where QPSK is combined with polarization modulation.

5 NON-IMAGING APPLICATIONS

The watermarking technique discussed here has potential applications to audio copyright protection and audio system and equalisation control. Two one-dimensional patterns can be embedded in each of the stereo channels on CD-ROM or DAT. These codes could designed to have a deliberately long span (such as GMW codes), in order to prevent cracking. These codes could also be employed in automatic spectral and delay calibration/equalisation of the sound system, because of their optimal impulse response. This feature could be particularly useful in dynamic situations, where the audio environment is constantly changing. A technique called Argent has been located on the internet as a commercial version of CD copyrighting, but so far, meaningful details on this method have been unavailable.

6 CONCLUSIONS

This paper demonstrates a method of encoding and recovery of a digital water mark on test images, using spread spectrum techniques. A critical analysis of the extension of the method to genuine two-dimensional patterns using non-binary characters is presented. The ultimate objective is the construction of an optimal set of colour patterns. A brief outline of the current state of the art is included.

7 ACKNOWLEDGEMENTS

The authors would like to extend their gratitude to Dr.Alisdair McAndrew, Nicholas Mee and Dr.Derek Rogers for the numerous helpful discussions on finite fields and coding theory.

[1] A.Z.Tirkel, G.A.Rankin, R.M.van Schyndel, W.J.Ho, N.R.A.Mee, C.F.Osborne, Electronic Water Mark. DICTA-93 Macquarie University, Sydney, December 1993. p.666-672.
[2] R.G. van Schyndel, A.Z.Tirkel, N.R.A.Mee, C.F.Osborne. A Digital Watermark. First IEEE Image Processing Conference, Houston TX, November 15-17, 1994, vol II, p.86-90.
[3] S.Walton. Image Authentication for a Slippery New Age. Dr.Dobb's Journal, April 1995. p.18-26, 82-87.
[4] F.M.Boland, J.K.K. Ó Rouanaidh and C.Dautzenberg. Watermarking Digital Images for Copyright Protection. In publication.
[5]. F.J. MacWilliams and N.J.A.Sloane. Pseudo-random Sequences and Arrays. Proc.IEEE, vol 64, 1715-1729, Dec.1976.
[6] S.W.Golomb and H.Taylor. Two-Dimensional Synchronization Patterns for Minimum Ambiguity. IEEE Trans. on Information Theory, vol IT-28, no.4, p.600-604, July 1982.
[8] R.G. van Schyndel, A.Z.Tirkel, C.F.Osborne. Towards a Robust Digital Watermark, ACCV'95 Conference, Nanyang Technological University, Singapore, December 5-8, 1995, vol 2, p.504-508
[9] I.S.Reed and R.M.Stewart. Note on the Existence of Perfect Maps. IRE Trans. on Information Theory. vol IT-8, p.10-12, Jan. 1962
[10] G.B. Rhoads. "Identification/Authentication Coding Method and Apparatus" Patent Application WO 95/14289.
[11] M.Cooperman. Digital Information Commodities Exchange (DICE). Argent Algorithm used by CANE Records (Independent Music Label run by University of Miami) Ref. ..arch.att.com/www-buyinfo/archive/95Q3/0084.html
[12] D.P.Rogers. "Non-Binary Spread Spectrum Multiple-Access Communications". Ph.D.Thesis, Department of Electrical and Electronic Engineering, University of Adelaide, March 1995.
[13] A.Z.Tirkel, R.G.van Schyndel, C.F.Osborne. "A Two Dimensional Digital Watermark", DICTA'95, University of Queensland, Brisbane, December 6-8, 1995. p.378-383.
[14] J.A.Lim, A.J.Maeder. "Image Authentication Extension for Lossy JPEG". DICTA'95, University of Queensland, Brisbane, December 6-8, 1995. p.210-216.
[15] D.H.Green "Structural properties of pseudorandom arrays and volumes and their related sequences" IEE Proceedings, Vol 132, Pt.E, No.3, May 1985, p.133-145.

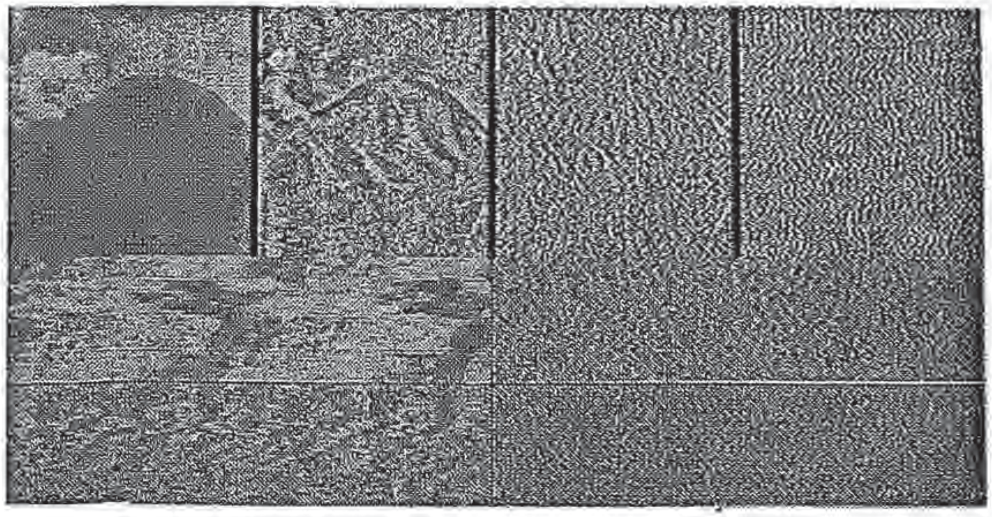
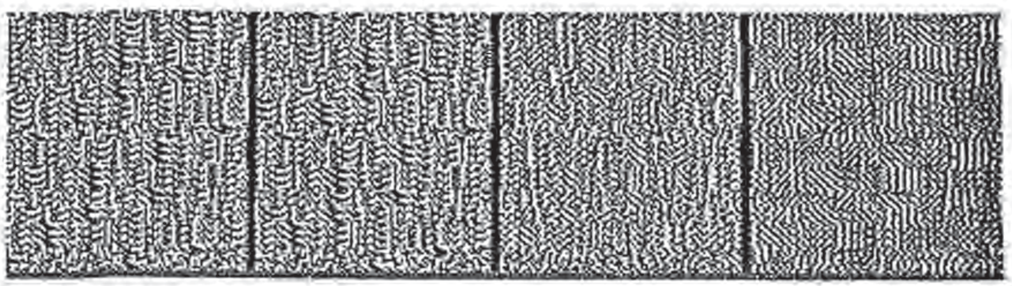


Figure 1

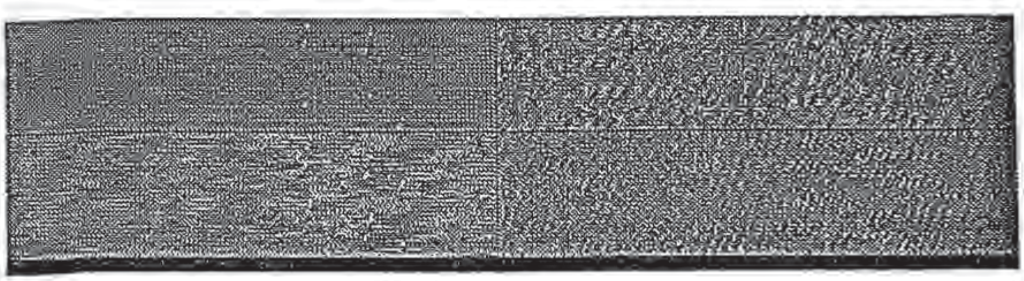
Upper (left to right: Encoded image after high pass filtering, removing
(a) 0, (b) 10, (c) 60, (d) 100 of 128 Spatial Frequency Components
Lower (Centre Left, Bottom Left, Centre Right, Bottom Right) : Corresponding Decoded Patterns
(Medium gray=0, darker=negative, lighter=positive - all image intensities have been suitably scaled)



BEST AVAILABLE COPY

Fig 2.

Correlation presented in line format (Watermark peaks are clearly visible)



IEE PROCEEDINGS

VISION, IMAGE AND SIGNAL PROCESSING

Volume 143, Number 4, August 1996

special section from IPA95

Watermarking digital images for copyright protection

J.J.K.Ó Ruanaidh
W.J.Dowling
F.M.Boland

Indexing terms: Copyright protection, Image processing, Steganography, Spread spectrum communications

Abstract: A watermark is an invisible mark placed on an image that is designed to identify both the source of an image as well as its intended recipient. The authors present an overview of watermarking techniques and demonstrate a solution to one of the key problems in image watermarking, namely how to hide robust invisible labels inside grey scale or colour digital images.

1 Introduction

Computers, printers and high rate transmission facilities are becoming less expensive and more generally available. It is now feasible and very economical to transmit images and video sequences using computer networks rather than to send hard copies by post. In addition, images may be stored in databases in digital form. A major impediment to the use of electronic distribution and storage is the ease of intercepting, copying and redistributing electronic images and documents in their *exact* original form. As a result, publishers are extremely reluctant to use this means of disseminating material. The commercial possibilities for the World Wide Web are steadily becoming more appreciated. However, if these possibilities are to be realised, an integrated approach to the secure handling, issue and duplication of issued documents is required. Public key encryption systems such as the RSA algorithm [1-3] do not completely solve the problem of unauthorised copying because of the ease with which images may be reproduced from previously published documents. All encrypted documents and images need to be decrypted before they can be inspected or used. Once encryption is removed the document can be passed on in an electronic form. If there is more than one recipient of an image, there is no direct proof that any particular authorised recipient is responsible for passing it on to unauthorised users. The idea of using an indelible watermark to identify uniquely both the source of an image and an intended recipient has therefore stimulated much interest in the electronic publishing and printing industries.

To be effective, an embedded watermark should be visually imperceptible, secure, reliable and resistant to attack.

Imperceptible. The image must not be visibly degraded by the presence of the mark. The mark should serve as a unique identifier with a high information content.

Secure and reliable. The mark must be strongly resistant to unauthorised detection and decoding. The watermark must also be capable of identifying the source and intended recipient with a low probability of error. It is also desirable that it would be difficult for an unauthorised agent to forge watermarks. Innovative error-control coding and digital signature techniques are required to ensure reliable and secure communication of the mark as well as authentication of the encoded message.

Robust. The mark must be robust to attack and must be tolerant to reasonable quality lossy compression of the image using transform coding, vector quantisation or any other technique. Standard image processing operations such as low pass filtering, cropping, translation and rescaling should not remove the mark.

Later we shall describe a method which fulfils most of the above requirements. In this paper, we argue that watermarking needs to be *adaptive* in order to be robust. In direct contrast to many other techniques, with the notable exception of Cox *et al.* [4], the method here places the watermark on the *most perceptually significant* components of an image. The logic behind the premise is quite simple. A watermark that is nonintrusive is one which resembles the image it is designed to protect. By virtue of its similarity to the image, any operation that is intentionally performed to damage the watermark will also damage the image.

The factors affecting the transmission of information embedded in images are quite complex. First, there is the need for robustness. The second factor is visibility. Intuitively, one can see that less information can be hidden on flat featureless regions of the image. It should be possible to incorporate more information into those parts of the image that contain more texture or around edges, provided edge integrity is maintained. Psychovisual phenomena are obviously factors in the transmission of hidden information.

There are two main principles involved in designing a watermark. The first principle, mentioned earlier, is

IEE Proceedings online no. 19960711
Paper first received 22nd December 1995 and in revised form 14th June 1996

J.J.K. Ó Ruanaidh was with Trinity College Dublin and is now with the Computer Vision Group, Centre Universitaire d'Informatique, 24 Rue Général Dufour, Université de Genève, CH 1211 Genève 4, Switzerland
W.J. Dowling and F.M. Boland are with the Department of Electronic and Electrical Engineering, Trinity College Dublin, Dublin 2, Ireland

that a successful watermarking algorithm should explicitly identify and place the mark in the most important features of the image. There are some similarities to the key ideas behind image compression and there will be many ideas and techniques borrowed from this field. The second principle, which we shall outline briefly, is that of *spread spectrum communications* [5].

2 Previous work

Brassil *et al.* [6] have investigated different methods for marking text within documents with a unique binary codeword which serves to identify legitimate users of the document. The codeword is embedded in a document by making subtle modifications to the structure of the document such as modulation of line width and interword spacing as well as modification of character fonts. The presence of the codeword does not visibly degrade the document but can be readily detected by making a comparison with the original. Standard document handing operations such as photocopying and scanning do not remove the mark. The same idea may be extended to include the protection of images.

Kurak and McHugh [7] have considered the possible application of redundant features in digital images to the transmission of information. Their concern was the transmission of dangerous viruses (or 'Trojan horse programs') in the least significant bits of a data stream. They note that merely viewing an image is not sufficient for detecting the presence of some form of corruption. Depending on the texture of the image and the quality of a computer monitor, it is possible to exploit the limited dynamic range of the human eye to hide low-quality images within other images. Walton [8] has developed a technique for introducing checksums in the least significant bits of an image to implement a fragile watermark and thus prevent unauthorised tampering. Dautzenberg and Boland [9] examined the use of the least significant bits as a possible scheme for introducing watermarks into images. This approach gave very poor results because standard lossy compression schemes, such as JPEG [10], tend to have the effect of randomising the least significant bits during the quantisation stage of image compression.

Zhao and Koch [11] have investigated an approach to watermarking images based on the JPEG [10] image compression algorithm. Their approach is to segment the image into individual 8×8 blocks. Only eight coefficients occupying particular positions in the 8×8 block of DCT coefficients can be marked. These comprise the low frequency components of the image block, but exclude the mean value coefficient (at coordinate (0,0)) as well as the low frequencies at coordinates (0,1) and (1,0). Three of the remaining DCT coefficients are selected using a pseudorandom number generator to convey information. The resemblance of this technique to frequency hop spread spectrum communications is mentioned by the authors [11]. Zhao and Koch also take the precaution of placing the blocks at random positions in the image in order to make a successful attack by an enemy less likely.

Tirkel *et al.* [12, 13] and van Schyndel *et al.* [14, 15] have applied the properties of *m*-sequences to produce watermarks that are resistant to filtering, image cropping and are reasonably robust to cryptographic attack. The original image is not required to decode the mark. Recent work [15] indicates progress towards producing more robust watermarks.

Matsui and Tanaka [16] have applied linear predictive coding for watermarking video, facsimile, dithered binary pictures and colour and grey scale images. Their approach to hiding a watermark is to make the watermark resemble quantisation noise. To a certain extent their approach can be considered to be perceptually adaptive because quantisation noise is concentrated around edges and textured features. Cox *et al.* [4] believe that this method may not be robust to cropping. Ó Ruanaidh *et al.* [17] and Cox *et al.* [4] have developed perceptually adaptive transform domain methods for watermarking. In direct contrast to the previous approaches listed above the emphasis was on embedding the watermark in the *most significant* components of an image. The general approach used in these papers is to divide the image into blocks. Each block is mapped into the transform domain using either the discrete cosine transform [10, 18–20], the Hadamard transform [1, 18] or the Daubechies wavelet transform [19]. Only the components that are most significant to image intelligibility are marked. A transform-based watermarking algorithm is described in more detail in Section 4.

Transform domain modulation schemes possess a number of desirable features. First, one can mark according to the perceptual significance of different transform domain components which means that one can adaptively place watermarks where they are least noticeable, such as within the texture of an image. As a result, a transform domain watermark tends to resemble the original image. The watermark is also irregularly distributed over the entire image sub-block which makes it more difficult for enemies in possession of independent copies of the image to decode and to read the mark.

The scheme described by Cox *et al.* [4] differs from that used by Ó Ruanaidh *et al.* [17] in several ways. The main differences lie in the detection and decoding of the mark. Cox *et al.* embed a unique Gaussian distributed sequence into the coefficients. The Gaussian distribution is chosen to prevent attacks by colluding parties comparing independent copies of the image. Ó Ruanaidh *et al.* employ an alternative approach whereby a binary code is directly embedded in the image. One advantage of the latter approach is that it avoids the need to maintain large databases of watermarks. A disadvantage is that the sequences thus produced are discrete valued and therefore the watermark is less resistant to colluding parties. However, there is nothing to prevent one from using continuous watermarks to convey digital information. This would combine the best features of both approaches.

The discrete Fourier transform (DFT) may also be used in watermarking. The discrete Fourier transform of a real image is generally complex valued. This leads to a magnitude and phase representation for the image. Transform domain methods described above mark the components of real valued transforms. Ó Ruanaidh *et al.* [21] and Ó Ruanaidh *et al.* [17] have also investigated the use of DFT phase for the transmission of information. There are a number of reasons for doing this. First and most importantly, the human visual system is far more sensitive to phase distortions than to magnitude distortions [22]. Oppenheim and Lim [23] investigated the relative importance of the phase and magnitude components of the DFT to the intelligibility of an image and found that phase is more significant.

own
oise

s the
form
and
very
iages.
each
r dec-
ermed
ay be
ed to
ve dis-
n of a
n com-
all, of
hat the
nponen-
ats this
e of the
side the
ethod of
to lossy
scanning

ising the
ocks, and
e and the
s around
size 256 ×
expect to
bits. This
plications.
edundancy
as well as
ill see, this
marking in

watermark-
dulation for
idly, a tech-
to be placed
d. Note that
exclusively
nsforms can-
in the paper
r watermark-

dapted from
h is a hybrid
ency shift key-

each pixel in

block so that

block.

5. Modulate selected coefficients of the transformation (e.g. using bidirectional coding). The coefficients that are selected are those that are *most* relevant to the intelligibility of the image.

6. Compute the inverse transform, denormalise, add the mean to each pixel in the block and replace the image block in the image.

Steps 2 and 3 above produce a normalised image sub-block with zero mean. Although one of the DCT coefficients computed in Step 4 already contains the mean, Step 2 is not redundant because the normalisation in Step 3 may only be carried out if the mean of the block is zero.

Watermark detection is easily performed by carrying out Steps 1 to 4 above on the original image and the watermarked image in parallel and comparing the values of the coefficients.

4.1 The number of bits

The most important factor in embedding a bit stream in an image is to determine the number of bits that can be placed into a given image block.

In a highly textured image block, energy tends to be more evenly distributed among the different DCT coefficients. In a flat featureless portion of the image the energy is concentrated in the low frequency components of the spectrum.

As stated earlier, the aim is to place more information bits where they are most robust to attack and are least noticeable. This may be accomplished by using a simple thresholding technique. The first stage is to use visual masking and to weight the transform coefficients $F(k_1, k_2)$, $0 \le k_1 < N_1$ and $0 \le k_2 < N_2$, according to a subjective measure of their visual perceptibility:

$$G(k_1, k_2) = w(k_1, k_2)F(k_1, k_2) \tag{1}$$

The most significant components are then selected by comparing the component magnitude squared to the total energy in the block. The coefficient $F(k_1, k_2)$ is selected if

$$|G(k_1, k_2)|^2 \ge \epsilon \sum_{k_1=0}^{N_1-1} \sum_{k_2=1}^{N_2-1} |G(k_1, k_2)|^2 \tag{2}$$

The quantisation tables [10] used in JPEG image compression can be exploited to choose the weighting in eqn. 1 for DCT watermarking with 8 × 8 blocks.

Lossy image compression algorithms are designed to disregard redundant information. Information bits placed within textured areas of the image are therefore more vulnerable to attack. There is a compromise to be reached between hiding a large number of information bits where they can least be seen, but where they can be attacked by image compression algorithms, or placing fewer bits on less textured but safer portions of the image. This may be achieved by opting for a moderately low value of threshold (e.g. $\epsilon = 0.2$).

It is worth noting that the number of bits that can be encoded using image transforms far exceeds that of the block-mean approach. The number of modulated DCT coefficients is generally around 10 000 for a typical image. In the case of Zhao and Koch's method, 3 bits of information are encoded into each 8 × 8 block. If the blocks are tiled over the image then one could obtain a maximum code rate of 3/64 bits/pixel.

It is important to note the differences between the aims in image compression and in watermarking images. In transform-based image compression, the goal is to obtain a small number of transform coefficients which can be used to obtain a good approximation to the original image. Small changes in the coefficient values should make little difference to the reconstructed image. However, the reverse does not necessarily hold since a small change to the image can result in a large change in the coefficient values (particularly when the basis images also change). This behaviour is obviously extremely undesirable since the embedded information depends on the value of these coefficients. The severity of this effect depends on the image transforms being used. Ill-conditioning tends to be much more severe for image transformations whose basis images are data-dependent (e.g. the singular value decomposition (SVD)). Image transformations with fixed basis functions (e.g. DCT and wavelet transforms) tend to exhibit more stable behaviour.

5 Reliable communications

The material in this paper thus far has described methods for watermarking images. However, we have not yet addressed the other main component in the watermarking problem, namely the reliable transmission of the watermark.

Reliable communication was proven by Shannon [25] to be theoretically possible providing the information rate does not exceed a threshold known as the channel capacity. In this Section we make some rather idealised assumptions regarding the form of the noise **n** corrupting a watermark and use information theory to derive rules for setting the optimal strength and location of the watermark **x**.

Let us write,

$$x_i + n_i = y_i \qquad 1 \leq i \leq N \tag{3}$$

where x_i is one element of a watermark vector of length N, n_i is an element of a *noise* vector and y_i is a element of a watermark distorted by image processing noise. All forms of image processing including vector quantisation, filtering and scanning introduce noise which degrades the watermark. We assume that the noise is additive, white, stationary and Gaussian:

$$p(y_i|x_i) = p(n_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y_i - x_i)^2}{2\sigma^2}\right] \tag{4}$$

We also assume that the n_i are uncorrelated and that

$$p(y_1, y_2 \cdots y_N | x_1, x_2 \cdots x_N) = \prod_{i=1}^{N} p(y_i|x_i) \tag{5}$$

Channel capacity [26] may be defined as

$$C = \max_{p(x)} I(\mathbf{X}; \mathbf{Y}) \tag{6}$$

where the watermark probability density function $p(x)$ is chosen to maximise the average mutual information $I(\mathbf{X}; \mathbf{Y})$.

According to Proakis [27] the capacity is maximised with respect to the distribution $p(x)$ if

$$p(x_i) = \frac{1}{\sqrt{2\pi\gamma^2}} \exp\left[-\frac{x_i^2}{2\gamma^2}\right] \tag{7}$$

which is a zero mean Gaussian density with variance γ^2. In this case,

$$I_{\max} = \frac{1}{2} N \log_2 \left[1 + \frac{\gamma^2}{\sigma^2}\right] \tag{8}$$

Note that eqn. 7 would seem to support the use of a Gaussian distributed watermark such as that used by Cox *et al.* [4].

In image watermarking we might expect that the transmission of information is functioning under quite extreme conditions, in which case $\sigma^2 >> \gamma^2$, which implies

$$\ln\left(1 + \frac{\gamma^2}{\sigma^2}\right) \approx \frac{\gamma^2}{\sigma^2} \tag{9}$$

Substituting the above into eqn. 8 we obtain the following condition for reliable communication:

$$\frac{\gamma^2}{\sigma^2} > (2 \ln 2) \frac{J}{N} \tag{10}$$

where the N is the number of sites used to hide watermark information bits and J is the information rate. Eqns. 8 and 10 reduce to the more familiar form [1] if the 'bandwidth' B of the channel is set to half the number of sites, $N/2$. Note that the noise power can be considerably greater than the signal power and, in theory at least, the message may still be transmitted reliably!

The strategy for communicating the watermark is now clear. Because a watermark should be imperceptible the signal to noise ratio (SNR) is severely limited. Reliable communication can only be assured by increasing bandwidth B to compensate for poor SNR. Hence, in the case of watermarking the maximum number N of suitable transform domain coefficients should be exploited for hiding information in the image. An analogous situation occurs in satellite and mobile communications where SNR is limited by power restrictions at the transmitter. There are also many similarities to secret military communications where an opponent may also attempt to detect, intercept or block a transmission. Watermarking may be considered as being an application of *spread spectrum communications* [5].

The Shannon limit may be approached by applying error control codes. Robust error correction techniques can be employed if necessary. Methods for error control coding are described by Sweeney [28], Chambers [1] and Blahut [29].

Information theory also gives some insights into where the watermark should be placed. Let us assume that the image may be considered as a collection of parallel uncorrelated Gaussian channels which satisfy eqn. 3 above with the constraint that the total watermark energy is limited:

$$\sum_{i=1}^{N} \gamma_i^2 \leq E \tag{11}$$

Using eqn. 4 and assuming that the noise variances are not necessarily the same in each channel, Gallager [26] shows that the capacity is

$$C = \frac{1}{2} \sum_{i=1}^{N} \log_2 \left(1 + \frac{\gamma_i^2}{\sigma_i^2}\right) \tag{12}$$

where σ_i^2 is the variance of the noise corrupting the watermark and γ_i^2 is the average power of the watermark signal in the ith channel. This is a more general form of eqn. 8. Capacity is achieved when

$$\gamma_i^2 + \sigma_i^2 = T_h \quad \text{if } \sigma_i^2 < T_h \tag{13}$$

$$\gamma_i^2 = 0 \quad \text{if } \sigma_i^2 \geq T_h \tag{14}$$

where the threshold T_h is chosen to maximise the sum on the left-hand side of eqn. 11 and thus maximise the energy of the watermark. This result shows clearly that the watermark should be placed in those areas where the local noise variance σ_i^2 is smaller than threshold T_N and not at all in those areas where the local noise variance exceeds the threshold. Note that the simple analysis presented here assumes that the noise corruption suffered by the watermark, as a result of common forms of image processing, is Gaussian. This is not an accurate assumption to make in many cases. However, the Gaussian assumption is not a bad choice given that the aim is to derive rules and heuristics that apply *in general* to a number of fundamentally different different image processing scenarios. The Gaussian noise model leads to a tractable analysis in many cases. Theoretically, it can also be considered to be a general noise model because of its conservative nature. Three justifications for its adoption in the absence of any information regarding the noise statistics include the central limit theorem [30], Herschel's theorem [31] as well as the principle of maximum entropy [32, 33]. In addition, additive white Gaussian noise theoretically gives the most difficult conditions in which to attempt communication [26]. Hence, the Gaussian noise assumption is actually quite conservative. A full analysis of the channel based on accurate knowledge of the noise statistics would lead to more accurate values for the channel capacity but would also be complicated by the need to evaluate difficult multidimensional integrals.

6 Examples

Fig. 1 shows 'Lena' watermarked using bidirectional coding and blocks with borders [9]. The image is of size 512 × 512 pixels, the inner block size is 12 × 12 pixels and the pixels are incremented by 3 to transmit a binary '1' and decremented by 3 to convey a binary '0'. The mark is for all intents and purposes invisible in Fig. 1 but may be detected quite readily [24, 9] even after lossy compression and scanning have been carried

Fig. 1 *Lena weakly watermarked using bidirectional coding*

Fig. 2 *Lena strongly watermarked using bidirectional coding*

Fig. 3 *Lena watermarked using fourth-order Daubechies wavelets*

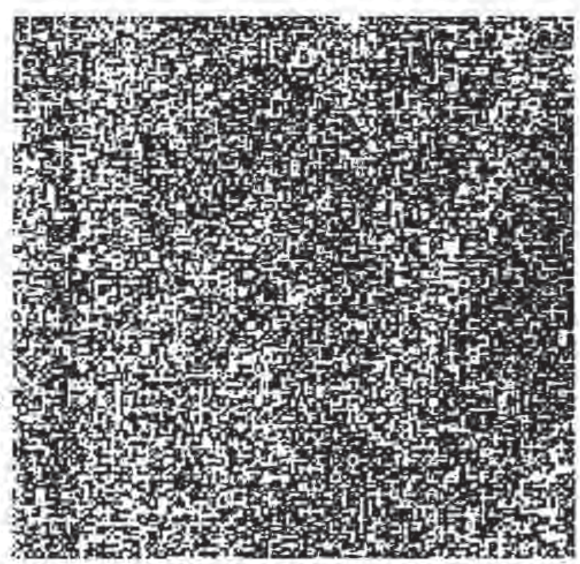


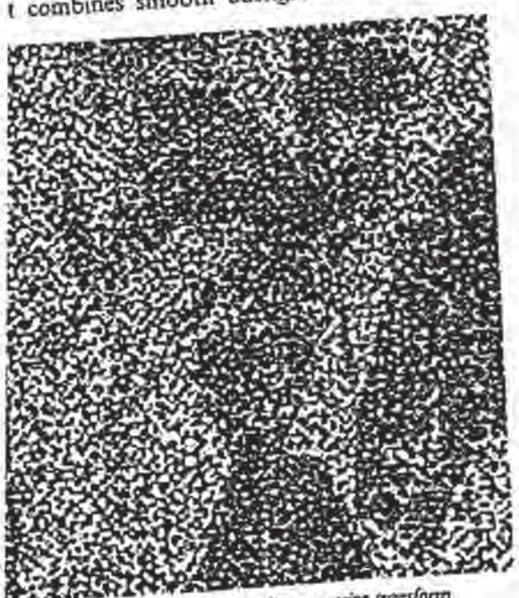
Fig. 4 *Watermark produced using Daubechies wavelets*

rmark conveys 441 bits of information in and the standard message reads: '012345 ermark...'. Fig. 2 shows the same image ced with a perturbation of ± 12 to make dily visible.

ws 'Lena' watermarked using the Daub- et transform. The block size is 8 × 8 and n transform coefficient perturbation is ± 5. ark is conveyed by modulating 15 551 coefficients. The standard message is arge number of times to occupy all of the pacity. Note that the presence of the mark io visible degradation.

nows the difference between the wavelet sion of the standard image and the original, factor of 30 and offset by 127 grey scale 5 shows a similar difference image for a produced using the DCT. As in the case of watermark, the DCT block size is 8 × 8 and um transform coefficient perturbation is ± 5. watermark is conveyed by modulating 11 933 coefficients and the standard test message is encoded as before.

hows a watermarked image of a wolf on a ckground. The image is of size 768 × 512. ge is very interesting from our point of view t combines smooth background regions (the



Watermark produced using the discrete cosine transform

.6 Marked image of a wolf on a snowy background

snowy ... watermark was produced using the Hadamard trans form with an energy threshold ε = 0.2. The block size is 8 × 8 and the transform coefficient perturbation is ± 10. The watermark is conveyed by modulating 3840 transform coefficients. The absolute difference between the original image and the marked image, contrast enhanced using histogram equalisation, is shown in Fig. 7. In this case, areas with high information density (expressed in terms of the number of embedded water- mark bits per block) are white, while areas which attract fewer watermark bits are darker. The outline of the wolf's head is quite clear. Note that, as before, information density is higher in textured regions.

Fig. 8 shows a segment of a watermarked image of Lena after JPEG [10] image compression followed by cropping. The size of the segment is 512 × 200 pixels. The watermark embedded in the uncropped image is 4096 bits long and the blocksize is 8 × 8 (i.e. just one bit per block). The encoded message consists of 32 bits ('0123' in ASCII). The watermark was placed using a DCT and the perturbation in the coefficient values was ± 10. JPEG was applied with a standard setting of 50 and no smoothing was used. By judicious use of con- catenated error control codes [29, 1, 28] the watermark was recovered with ease from this cropped section.

It is apparent upon examining the watermarks in Figs. 4, 5 and 7 that the transform-based marking schemes possess a number of desirable features. One can mark according to the distribution of energy within the coefficients. In this way, one can place watermarks where they are least noticeable, such as within image texture and around edges. As a result, the watermark exhibits a ghost-like resemblance to the original image.

Fig.7 Watermark around the region of the wolf's head
The watermark was generated using the Hadamard transform. Areas with a high density of information are indicated by the brighter blocks

Fig.8 Cropped grey scale image of Lena
The size of image is 512 × 200 pixels

...onclusion

paper has outlined a scheme for embedding robust ...marks in digital images. The watermarks are ...ned to be invisible, even to a careful observer, but ...ain sufficient information to identify both the ori... ...nd intended recipient of an image with a very low ...ability of error.

...e key feature of the transform-based methods is ...information bits can be placed adaptively, thereby ...ing the watermark more robust to attack. A water... ...k is made imperceptible because it is designed to ...ch the characteristics of the image to be protected. ...nsform-based methods have proven to be reasona... robust to image compression and standard image ...cessing operations. In addition, transform-based ...hods yield a relatively large number of transform ...fficients in which to embed the watermark. Future ...k will include the use of human visual models in ...gning watermarking schemes. The application of ...able error correction codes and digital signature ...niques will also be investigated. In particular, the ...istical characteristics of the *watermarking channel* ...d careful study. It is known that the distribution of ...DCT coefficients of a typical image is well approxi... ...ted by a Laplacian distribution [18]. It has been ...served that the noise distortion imposed on the ...termark by common image processing operations is ...n-Gaussian and impulsive in nature. Soft error con... ...l codes designed for additive wideband Gaussian ...ise (AWGN) channels (e.g. Reed–Muller codes) are ...t particularly effective in this application. The design ...an optimal detector for the watermark depends on a ...ar knowledge of the noise statistics because such a ...tector can only be as good as the model assumptions ...on which it is based. Finally, work will continue on ...vising watermarking schemes that do not require the ...iginal image to decode the watermark [21].

Acknowledgment

...his work was supported by a Forbairt strategic ...search grant. The authors would like to thank Dr ...eter J. Cullen for helpful advice and stimulating dis... ...ussions.

References

CHAMBERS, W.G.: 'Basics of communications and coding' (Oxford Science Publications, Clarendon Press, Oxford, 1985)
HAYKIN, S.: 'Communications systems' (Wiley, 1994, 3rd edn.)
SCHNEIER, B.: 'Applied cryptography' (Wiley, 1995, 2nd edn.)
COX, I., KILLIAN, J., LEIGHTON, T., and SHAMOON, T.: 'Secure spread spectrum communication for multimedia'. Technical report, NEC Research Institute, 1995. ftp://ftp.nj.nec.com/pub/ingemar/papers/watermark.ps.Z
5 PICKHOLTZ, R.L., SCHILLING, D.L., and MILSTEIN, L.B.: 'Theory of spread spectrum communications-a tutorial', *IEEE Trans.*, 1982, COM-30, (5), pp. 855–884

... ...age document copying'. Proceedings of INFOCOM 94, 1994
7 KURAK, C., and McHUGH, J.: 'A cautionary note on image downgrading'. Proceedings 8th Annual Computer Security Applications Conference, San Antonio, 1992
8 WALTON, S.: 'Image authentication for a slippery new age', *Dr. Dobb's J.*, 1995, 20, (4), pp. 18–26, 82–87
9 DAUTZENBERG, C., and BOLAND, F.M.: 'Watermarking images'. Technical report, Department of Electronic and Electrical Engineering, Trinity College Dublin, 1994
10 PENNEBAKER, W.B., and MITCHELL, J.L.: 'JPEG still image compression standard' (Van Nostrand Reinhold, New York, 1993)
11 ZHAO, J., and KOCH, E.: 'Embedding robust labels into images for copyright protection'. Technical report, Fraunhofer Institute for Computer Graphics, Darmstadt, Germany, 1994
12 TIRKEL, A.Z., RANKIN, G.A., VAN SCHYNDEL, R.G., HO, W.J., MEE, N.R.A., and OSBORNE, C.F.: 'Electronic water mark'. Proceedings of Dicta-93, 1993, pp. 666–672
13 TIRKEL, A.Z., VAN SCHYNDEL, R.G., and OSBORNE, C.F.: 'A two-dimensional digital watermark'. Proceedings of ACCV, Singapore, 1995
14 VAN SCHYNDEL, R.G., TIRKEL, A.Z., and OSBORNE, C.F.: 'A digital watermark'. Proceedings of IEEE International Conference on *Image Processing*, Austin, Texas, 1994, pp. 86–90
15 VAN SCHYNDEL, R.G., TIRKEL, A.Z., and OSBORNE, C.F.: 'Towards a robust digital watermark'. Proceedings of Dicta-95, 1995
16 MATSUI, K., and TANAKA, K.: 'Video-steganography: how to secretly embed a signature in a picture'. IMA Intellectual Property Project Proceedings, January 1994, pp. 187–206
17 Ó RUANAIDH, J.J.K., DOWLING, W.J., and BOLAND, F.M.: 'Phase watermarking of images'. IEEE International Conference on *Image processing*, Lausanne, Switzerland, September 1996
18 CLARKE, R.J.: 'Transform coding of images' (Academic Press, London, 1985)
19 PRESS, W.H., TEUKOLSKY, S.A., VETTERLING, W.T., and FLANNERY, B.P.: 'Numerical recipes in C' (Cambridge University Press, 1992, 2nd edn.)
20 RAO, K.R., and YIP, P.: 'The discrete cosine transform: algorithms, advantages, applications' (Academic Press, 1990)
21 Ó RUANAIDH, J.J.K., BOLAND, F.M., and SINNEN, O.: 'Watermarking digital images for copyright protection'. Proceedings of the Electronic Imaging and Visual Arts Conference, Florence, February 1996. http://kalman.mee.tcd.ie/people/jjr/eva-pap.html
22 LIM, J.S.: 'Two-dimensional signal and image processing' (Prentice-Hall International, 1990)
23 OPPENHEIM, A.V. and LIM J.S.: 'The importance of phase in signals', *Proc. IEEE*, 1981, 69, (5), pp. 529–541
24 CARONNI, G.: 'Assuring ownership rights for digital images', *in* BRUEGGEMANN, H.H., and GERHARDT-HAECKL, W. (Eds): 'Reliable IT systems VIS '95' (Vieweg Publishing Company, Germany, 1995)
25 SHANNON, C.E.: 'A mathematical theory of communication', *Bell Syst. Tech. J.*, 1948, 27, pp. 379–423, 623–656
26 GALLAGER, R.G.: 'Information theory and reliable communication' (Wiley, 1968)
27 PROAKIS, J.G.: 'Digital communication'. Series in electrical and computer engineering communications and signal processing (McGraw-Hill, 1995, 3rd edn.)
28 SWEENEY, P.: 'Error control coding: an introduction' (Prentice-Hall, 1991)
29 BLAHUT, R.E.: 'The theory and practice of error control codes' (Addison-Wesley, 1983)
30 PAPOULIS, A.: 'Probability, random variables and stochastic processes' (McGraw-Hill, 1984, 2nd edn.)
31 BRETTHORST, G.L.: 'Bayesian spectrum analysis and parameter estimation' (Springer-Verlag, 1989)
32 JAYNES, E.T.: '', *in* ROSENKRANTZ, R.D. (Ed.): 'Papers on probability, statistics and statistical physics, a reprint collection' (Kluwer, 1989)
33 BURTON, D., and FITZGERALD, W.J.: 'Bayesian parameter estimation: further results'. Technical report, Marconi Maritime Research Laboratory, Cambridge, England, 1989

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☒ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

©NEC Research Institute, Technical Report 95 – 10.

Secure Spread Spectrum Watermarking for Multimedia

Ingemar J. Cox†, Joe Kilian†, Tom Leighton‡ and Talal Shamoon†*

Abstract

We describe a digital watermarking method for use in audio, image, video and multimedia data. We argue that a watermark must be placed in perceptually significant components of a signal if it is to be robust to common signal distortions and malicious attack. However, it is well known that modification of these components can lead to perceptual degradation of the signal. To avoid this, we propose to insert a watermark into the spectral components of the data using techniques analogous to spread spectrum communications, hiding a narrow band signal in a wideband channel that is the data. The watermark is difficult for an attacker to remove, even when several individuals conspire together with independently watermarked copies of the data. It is also robust to common signal and geometric distortions such as digital-to-analog and analog-to-digital conversion, resampling, and requantization, including dithering and recompression and rotation, translation, cropping and scaling. The same digital watermarking algorithm can be applied to all three media under consideration with only minor modifications, making it especially appropriate for multimedia products. Retrieval of the watermark unambiguously identifies the owner, and the watermark can be constructed to make counterfeiting almost impossible. Experimental results are presented to support these claims.

1 Introduction

The proliferation of digitized media (audio, image and video) is creating a pressing need for copyright enforcement schemes that protect copyright ownership. Conventional cryptographic systems permit only valid keyholders access to encrypted data, but once such data is decrypted there is no way to track its reproduction or retransmission. Conventional cryptography therefore provides little protection against data piracy, in which a publisher is confronted with unauthorized reproduction of information. A digital watermark is intended to complement cryptographic processes. It is a visible, or preferably invisible, identification code that is permanently embedded in the data, that is, it remains present within the data after any decryption process. In the context of this work, data refers to audio (speech and music), images (photographs and graphics), and video (movies). It does not include ASCII representations of text, but does include text

[0]†Post: NEC Research Institute, 4 Independence Way, Princeton, NJ 08540.
Email: ingemar|joe|talal@research.nj.nec.com
[0]‡Post:Mathematics Department and Laboratory for Computer Science, MIT, Cambridge, MA 02139.
Email: ftl@math.mit.edu
*Authors appear in alphabetical order.

represented as an image. A simple example of a digital watermark would be a visible "seal" placed over an image to identify the copyright owner. However, the watermark might contain additional information, including the identity of the purchaser of a particular copy of the material.

In order to be effective, a watermark should be:

Unobtrusive The watermark should be perceptually invisible, or its presence should not interfere with the work being protected.

Robust The watermark must be difficult (hopefully impossible) to remove. Of course, in theory, any watermark may be removed with sufficient knowledge of the process of insertion. However, if only partial knowledge is available, for example, the exact location of the watermark within an image is unknown, then attempts to remove or destroy a watermark by say, adding noise, should result in severe degradation in data fidelity before the watermark is lost. In particular, the watermark should be robust to

- **Common signal processing** The watermark should still be retrievable even if common signal processing operations are applied to the data. These include, digital-to-analog and analog-to-digital conversion, resampling, requantization (including dithering and recompression), and common signal enhancements to image contrast and color, or audio bass and treble, for example.

- **Common geometric distortions (image and video data)** Watermarks in image and video data should also be immune from geometric image operations such as rotation, translation, cropping and scaling.

- **Subterfuge Attacks: Collusion and Forgery** In addition, the watermark should be robust to collusion by multiple individuals who each possess a watermarked copy of the data. That is, the watermark should be robust to combining copies of the same data set to destroy the watermarks. Further, if a digital watermark is to be used as evidence in a court of law, it must not be possible for colluders to combine their images to generate a different valid watermark with the intention of framing a third-party.

Universal The same digital watermark algorithm should apply to all three media under consideration. This is potentially helpful in the watermarking of multimedia products. Also, this feature is conducive to implementation of audio and image/video watermarking algorithms on common hardware.

Unambiguous Retrieval of the watermark should unambiguously identify the owner. Further, the accuracy of owner identification should degrade gracefully in the face of attack.

Previous digital watermarking techniques, described in Section 2, are not robust, and the watermark is easy to remove. In addition, it is unlikely that any of the earlier watermarking methods would survive common signal and geometric distortions. The principal reason for these weaknesses is that previous methods have not explicitly identified the perceptually most significant components of a signal as the destination for the watermark. In fact, it is often the case that the perceptually significant regions are explicitly avoided. The reason for this is obvious - modification of perceptually significant components of a signal results in perceptual distortions much earlier than if the modifications are applied to perceptually insignificant regions. Hence, for example, the common stategy of placing a watermark in the high frequency components of a signal's spectrum.

The key insight of this paper is that in order for it to be robust, the watermark *must* be placed in perceptually significant regions of the data despite the risk of potential fidelity distortions. Conversely, if the watermark is placed in perceptually insignificant regions, it is easily removed, either intentionally or unintentionally by, for example, signal compression techniques that implicitly recognize that perceptually weak components of a signal need not be represented.

The perceptually significant regions of a signal may vary depending on the particular media (audio, image or video) at hand, and even within a given media. For example, it is well known that the human visual system is tuned to certain spatial frequencies and to particular spatial characteristics such as line and corner features. Consequently, many watermarking schemes that focus on different phenomena that are perceptually significant are potentially possible. In this paper, we focus on perceptually significant *spectral* components of a signal.

Section 3 begins with a discussion of how common signal transformations, such as compression, quantization and manipulation, affect the frequency spectrum of a signal. This motivates why we believe that a watermark should be embedded in the data's perceptually significant frequency components. Of course, the major problem then becomes how to insert a watermark into perceptually significant components of the frequency spectrum without introducing visible or audible distortions. Section 3.2 proposes a solution based on ideas from spread spectrum communications.

The structure of a watermark may be arbitrary. However, Section 4 provides an analysis based on possible collusion attacks that indicates that a binary watermark is not as robust as a continuous one. Furthermore, we show that a watermark structure based on sampling drawn from multiple i.i.d Gaussian random variables offers good protection against collusion.

Of course, no watermarking system can be made perfect. For example, a watermark placed in a textual image may be eliminated by using optical character recognition technology. However, for common signal and geometric distortions, the experimental results of Section 5 strongly suggest that our system satisfies *all* of the properties discussed in the introduction, and displays strong immunity to a wide variety of attacks, though more extensive experiments are needed to confirm this. Finally, Section 6 discusses possible weaknesses and enhancements to the system.

2 Previous Work

Several previous digital watermarking methods have been proposed. L. F. Turner [Tur89] proposed a method for inserting an identification string into a digital audio signal by substituting the "insignificant" bits of randomly selected audio samples with the bits of an identification code. Bits are deemed "insignificant" if their alteration is inaudible. Such a system is also appropriate for two dimensional data such as images, as discussed in [vSTO94]. Unfortunately, Turner's method may easily be circumvented. For example, if it is known that the algorithm only affects the least significant two bits of a word, then it is possible to randomly flip *all* such bits, thereby destroying any existing identification code.

Caronni [Car95] suggests adding *tags* — small geometric patterns – to digitized images at brightness levels that are imperceptable. While the idea of hiding a spatial watermark in an image is fundamentally sound, this scheme is susceptible to attack by filtering and redigitization. The fainter such watermarks are the more susceptible they are such attacks and geometric shapes provide only a limited alphabet with which to encode information. Moreover, the scheme is not applicable to audio data and may not be robust to common geometric distortions, especially cropping.

Brassil *et al* [BLMO94] propose three methods appropriate for document images in which text is common. Digital watermarks are coded by: (1) vertically shifting text lines, (2) horizontally shifting words, or (3) altering text features such as the vertical endlines of individual characters. Unfortunately, all three proposals are easily defeated, as discussed by the authors. Moreover, these techniques are restricted exclusively to images containing text.

Tanaka *et al* [TNM90, MT94] describe several watermarking schemes that rely on embedding watermarks that resemble quantization noise. Their ideas hinge on the notion that quantization noise is typically imperceptible to viewers. Their first scheme injects a watermark into an image by using a predetermined data stream to guide level selection in a predictive quantizer. The data stream is chosen so that the resulting image looks like quantization noise. A variation on this scheme is also presented, where a watermark in the form of a dithering matrix is used to dither an image in a certain way. There are several drawbacks to these schemes. The most important is that they are susceptible to signal processing, especially requantization, and geometric attacks such as cropping. Furthermore, they degrade an image in the same way that predictive coding and dithering can.

In [TNM90], the authors also propose a scheme for watermarking facsimile data. This scheme shortens or lengthens certain runs of data in the run length code used to generate the coded fax image. This proposal is susceptible to digital-to-analog and analog-to-digital attacks. In particular, randomizing the LSB of each pixel's intensity will completely alter the resulting run length encoding. Tanaka *et al* also propose a watermarking method for "color-scaled picture and video sequences". This method applies the same signal transform as JPEG (DCT of 8×8 sub-blocks of an image) and embeds a watermark in the coefficient quantization module. While being compatible with existing transform coders, this scheme is quite susceptible to requantization and filtering and is equivalent to coding the watermark in the least significant bits of the transform coefficients.

In a recent paper, Macq and Quisquater [MQ95] briefly discuss the issue of watermarking digital images as part of a general survey on cryptography and digital television. The authors provide a description of a procedure to insert a watermark into the least significant bits of pixels located in the vicinity of image contours. Since it relies on modifications of the least significant bits, the watermark is easily destroyed. Further, their method is restricted to images, in that it seeks to insert the watermark into image regions that lie on the edge of contours.

Bender *et al* [BGM95] describe two watermarking schemes. The first is a statistical method called "Patchwork" that somewhat resembles the statistical component of our proposal. Patchwork randomly chooses n pairs of image points, (a_i, b_i), and increases the brightness at a_i by one unit while correspondingly decreasing the brightness of b_i. The expected value of the sum of the differences of the n pairs of points is then claimed to be $2n$, provided certain statistical properties of the image are true. In particular, it is assumed that all brightness levels are equally likely, that is, intensities are uniformly distributed. However, in practice, this is very uncommon. Moreover, the scheme may (1) not be robust to randomly jittering the intensity levels by a single unit, and (2) be extremely sensitive to geometric affine transformations.

The second method is called "texture block coding", wherein a region of random texture pattern found in the image is copied to an area of the image with similar texture. Autocorrelation is then used to recover each texture region. The most significant problem with this technique is that it is only appropriate for images that possess large areas of random texture. The technique could not be used on images of text, for example. Nor is there a direct analog for audio.

Digimarc Corporation or Portland, Oregon, describe a method that adds or subtracts small random quantities from each pixel. Addition or subtraction is determined by comparing a binary mask of L bits with the LSB of each pixel. If the LSB is equal to the corresponding mask bit, then the random quantity is added, otherwise it is subtracted. The watermark is subtracted by first computing the difference between the original and watermarked images and then by examining the sign of the difference, pixel by pixel, to determine if it corresponds to the original sequence of additions and subtractions. The Digimarc method does not make use of perceptual relevance and is probably equivalent to adding high frequency noise to the image. As such, it may not be robust to low pass filtering.

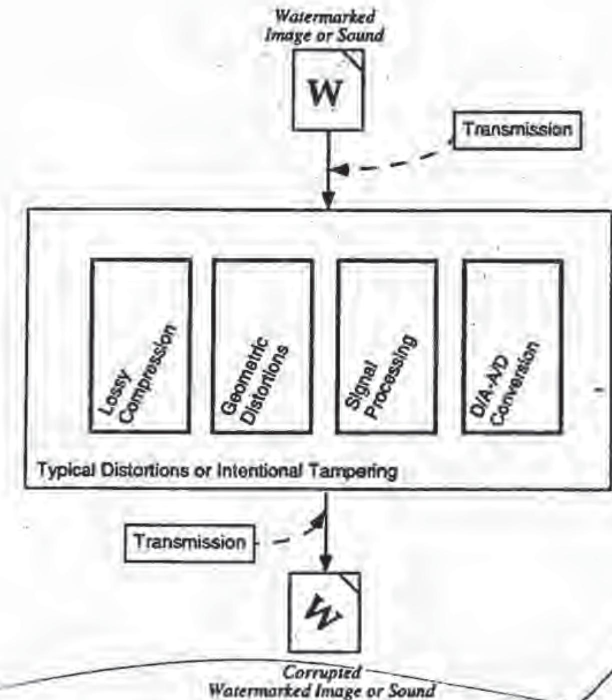
Koch, Rindfrey and Zhao [KRZ94] propose two general methods for watermarking images. The first method, attributed to Scott Burgett, breaks up an image into 8×8 blocks and computes the Discrete Cosine Transform (DCT) of each of these blocks. A pseudorandom subset of the blocks is chosen, then, in each such block, a triple of frequencies is selected from one of 18 predetermined triples and modified so that their relative strengths encode a 1 or 0 value. The 18 possible triples are composed by selection of three out of eight predetermined frequencies within the 8×8 DCT block. The choice of the 8 frequencies to be altered within the DCT block is based on a belief that the "middle frequencies ... have moderate variance", i.e. they have similar magnitude. This property is needed in order to allow the relative strength of the frequency triples to be altered without requiring a modification that would be perceptually noticeable. Superficially, this scheme is similar to our own proposal and, in fact, also draws analogy with spread spectrum communication. However, the structure of their watermark is different from ours. The set of frequencies is not chosen based on any perceptual significance or relative energy considerations. Further, because the variance between the eight frequency coefficients is small, one would expect that their technique may be sensitive to noise or distortions. This is supported by the experimental results which report that the "embedded labels are robust against JPEG compression for a quality factor as low as about 50%". By comparison, we demonstrate that our method performs well with compression quality factors as low as 5%. An earlier proposal by Koch and Zhao [KZ95] used not triples of frequencies but pairs of frequencies, and was again designed specifically for robustness to JPEG compression. Nevertheless, they state that " a lower quality factor will increase the likelihood that the changes necessary to superimpose the embedded code on the signal will be noticeably visible".

In a second method, designed for black and white images, no frequency transform is employed. Instead, the selected blocks are modified so that the relative frequency of white and black pixels encodes the final value. Both watermarking procedures are particularly vulnerable to multiple document attacks. To protect against this, Zhao and Koch propose a *distributed* 8×8 block created by randomly sampling 64 pixels from the image. However, the resulting DCT has no relationship to that of the true image and consequently may be likely to cause noticeable artifacts in the image and be sensitive to noise.

In addition to direct work on watermarking images, there are several works of interest in related areas. Adelson [Ade90] describes a technique for embedding digital information in an analog signal for the purpose of inserting digital data into an analog TV signal. The analog signal is quantized into one of two disjoint ranges, ($\{0, 2, 4 \ldots\}$,$\{1, 3, 5 \ldots\}$, for example) which are selected based on the binary digit to be transmitted. Thus Adelson's method is equivalent to watermark schemes that encode information into the least significant bits of the data or its transform coefficients. Adelson recognizes that the method is susceptible to noise and therefore proposes an alternative scheme wherein a 2×1 Hadamard transform of the digitized analog signal is taken. The differential coefficient of the Hadamard transform is offset by 0 or 1 unit prior to computing the inverse transform. This corresponds to encoding the watermark into the least significant bit of the differential coefficient of the Hadamard transform. It is not clear that this approach would demonstrate enhanced resilience to noise. Furthermore, like all such least significant bit schemes, an attacker can eliminate the watermark by randomization.

Schreiber *et al* [SLAN91] describe a method to interleave a standard NTSC signal within an enhanced definition television (EDTV) signal. This is accomplished by analyzing the frequency spectrum of the EDTV signal (larger than that of the NTSC signal) and decomposing it into three sub-bands (L,M,H for low, medium and high frequency respectively). In contrast, the NTSC signal is decomposed into two subbands, L and M. The coefficients, M_k, within the M band are quantized into m levels and the high frequency coefficients, H_k, of the EDTV signal are scaled such that the addition of the H_k signal plus any noise present in the system is less than the minimum separation between quantization levels. Once more, the method relies on modifying least significant bits. Presumably, the mid-range rather than low frequencies were chosen because these are less perceptually significant. In contrast, the method proposed here modifies the *most* perceptually significant components of the signal.

Finally, it should be noted that many, if not all, of the prior art protocols are not collusion resistant.



only Frequ

Figure 1: Common processing operations that a media document could undergo

3 Watermarking in the Frequency Domain

In this section, we first discuss how common signal distortion affect the frequency spectrum of a signal. This analysis supports our contention that a watermark must be placed in perceptually significant regions of a signal if it is to be robust. Section 3.2 proposes inserting a watermark into the perceptually most significant components of the spectrum using spread spectrum techniques.

3.1 Common signal distortions and their effect on the frequency spectrum of a signal

In order to understand the advantages of a frequency-based method, it is instructive to examine the processing stages that an image (or sound) may undergo in the process of copying, and to study the effect that these stages could have on the data, as illustrated in Figure 1. In the figure, "transmission" refers to the application of any source or channel code, and/or standard encryption technique to the data. While most of these steps are information lossless, many compression schemes (JPEG, MPEG etc.) can potentially degrade the data's quality, through *irretrievable* loss of data. In general, a watermarking scheme should be resilient to the distortions introduced by such algorithms.

Lossy compression is an operation that usually eliminates perceptually non-salient components of an image or sound. If one wishes to preserve a watermark in the face of such an operation, the watermark must be placed in the perceptually significant regions of the data. Most processing of this sort takes place in the frequency domain. In fact, data loss usually occurs among the high frequency components. Hence, the watermark must be placed in the *significant* frequency components of the image (or sound) spectrum.

After receipt, an image may endure many common transformations that are broadly categorized as geometric distortions or signal distortions. Geometric distortions are specific to images and video, and include such operations as rotation, translation, scaling and cropping. By manually determining a minimum of four or nine corresponding points between the original and the distorted watermark, it is possible to remove any two or three dimensional affine transformation [Fau93]. However, an affine scaling (shrinking) of the image leads to a loss of data in the high frequency spectral regions of the image. Cropping, or the cutting out and removal of portions of an image, also leads to irretrievable loss of data. Cropping may be a serious threat to any spatially based watermark such as [Car95] but is less likely to affect a frequency-based scheme, as shown in Section 5.5.

Common signal distortions include digital-to-analog and analog-to-digital conversion, resampling, re-quantization, including dithering and recompression, and common signal enhancements to image contrast and/or color, and audio frequency equalization. Many of these distortions are non-linear, and it is difficult to analyze their effect in either a spatial or frequency based method. However, the fact that the original image is known allows many signal transformations to be undone, at least approximately. For example, histogram equalization, a common non-linear contrast enhancement method, may be removed substantially by histogram specification [GW93] or dynamic histogram warping [CRH95] techniques.

Finally, the copied image may not remain in digital form. Instead, it is likely to be printed, or an analog recording made (onto analog audio or video tape). These reproductions introduce additional degradation into the image that a watermarking scheme must be robust to.

The watermark must not only be resistant to the inadvertant application of the aforementioned distortions. It must also be immune to intentional manipulation by malicious parties. These manipulations can include combinations of the above distortions, and can also include collusion and forgery attacks.

3.2 Spread spectrum coding of a watermark

The above discussion makes it clear that the watermark should *not* be placed in perceptually insignificant regions of the image or its spectrum since many common signal and geometric processes affect these components. For example, a watermark placed in the high frequency spectrum of an image can be easily eliminated with little degradation to the image by any process that directly or indirectly performs low pass filtering. The problem then becomes how to insert a watermark into the most perceptually significant regions of an spectrum without such alterations becoming noticeable. Clearly, any spectral coefficient may be altered, provided such modification is small. However, very small changes are very susceptible to noise.

To solve this problem, the frequency domain of the image or sound at hand is viewed as a *communication channel*, and correspondingly, the watermark is viewed as a signal that is transmitted through it. Attacks and unintentional signal distortions are thus treated as noise that the immersed signal must be immune to. While we use this methodology to hide watermarks in data, the same rationale can be applied to sending any type of message through media data.

Rather than encode the watermark into the least significant components of the data, we originally conceived our approach by analogy to spread spectrum communications [PSM82]. In spread spectrum communications, one transmits a narrowband signal over a much larger bandwidth such that the signal energy present in any single frequency is imperceptible. Similarly, the watermark is spread over very many frequency bins so that the energy in any one bin is very small and certainly undetectable. Nevertheless, because the watermark verification process knows of the location and content of the watermark, it is possible to concentrate these many weak signals into a single signal with high signal-to-noise ratio. However, to destroy such a watermark would require noise of high amplitude to be added to *all* frequency bins.

Spreading the watermark throughout the spectrum of an image ensures a large measure of security against unintentional or intentional attack: First, the location of the watermark is not obvious. Furthermore, frequency regions should be selected in a fashion that ensures severe degradation of the original data following any attack on the watermark.

A watermark that is well placed in the frequency domain of an image or a sound track will be practically impossible to see or hear. This will always be the case if the energy in the watermark is sufficiently small in any single frequency coefficient. Moreover, it is possible to increase the energy present in particular frequencies by exploiting knowledge of masking phenomena in the human auditory and visual systems. Perceptual masking refers to any situation where information in certain regions of an image or a sound is occluded by perceptually more prominent information in another part of the scene. In digital waveform coding, this frequency domain (and, in some cases, time/pixel domain) masking is exploited extensively to achieve low bit rate encoding of data [JJS93, GG92]. It is clear that both the auditory and visual systems attach more resolution to the high energy, low frequency, spectral regions of an auditory or visual scene [JJS93]. Further, spectrum analysis of images and sounds reveals that most of the information in such data is located in the low frequency regions.

Figure 2 illustrates the general procedure for frequency domain watermarking. Upon applying a frequency transformation to the data, a *perceptual mask* is computed that highlights perceptually significant regions in the spectrum that can support the watermark without affecting perceptual fidelity. The watermark signal is then inserted into these regions in a manner described in Section 4.2. The precise magnitude of each modification is only known to the owner. By contrast, an attacker may only have knowledge of the possible range of modification. To be confident of eliminating a watermark, an attacker must assume that each modification was at the limit of this range, despite the fact that few such modifications are typically this large. As a result, an attack creates visible (or audible) defects in the data. Similarly, unintentional signal distortions due to compression or image manipulation, must leave the perceptually significant spectral components intact, otherwise the resulting image will be severely degraded. This is why the watermark is robust.

In principle, any frequency domain transform can be used. However, for the experimental results of Section 5 we use a Fourier domain method based on the discrete cosine transform (DCT) [Lim90], although we are currently exploring the use of wavelet-based schemes as a variation. In our view, each coefficient in the frequency domain has a *perceptual capacity*, that is, a quantity of additional information can be added without any (or with minimal) impact to the perceptual fidelity of the data. To determine the perceptual capacity of each frequency, one can use models for the appropriate perceptual system or simple experimentation.

In practice, in order to place a length n watermark into an $N \times N$ image, we computed the $N \times N$ DCT of the image and placed the watermark into the n highest magnitude coefficients of the transform matrix, excluding the DC component.[1] For most images, these coefficients will be the ones corresponding to the low frequencies. Reiterating, the purpose of placing the watermark in these locations is because significant tampering with these frequency will destroy the image fidelity well before the watermark.

In the next section, we provide a high level discussion of the watermarking procedure, describing the

[1]More generally, n randomly chosen coefficients could be chosen from the $M, M \geq n$ most perceptually significant coefficients of the transform.

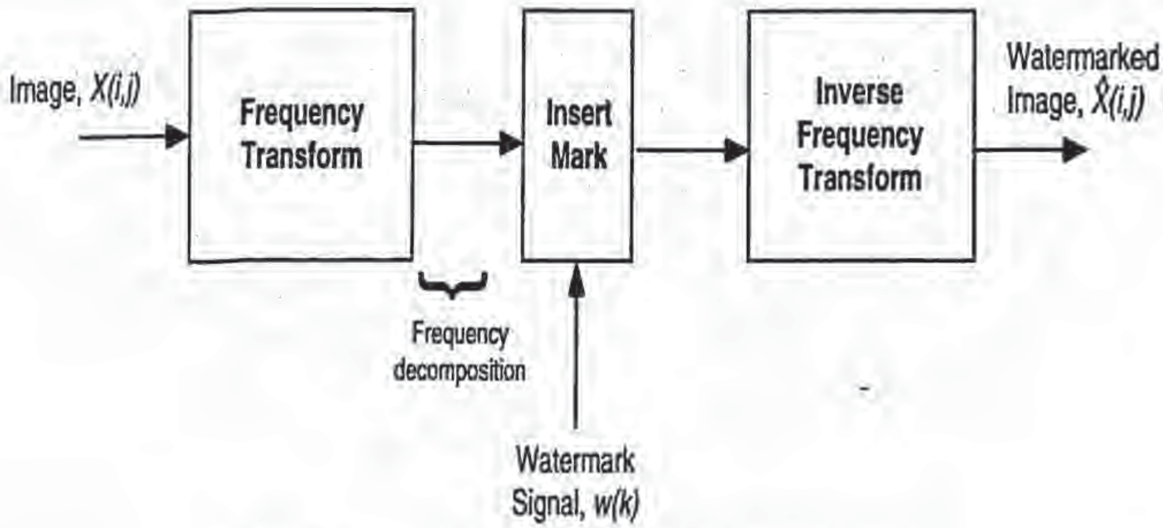
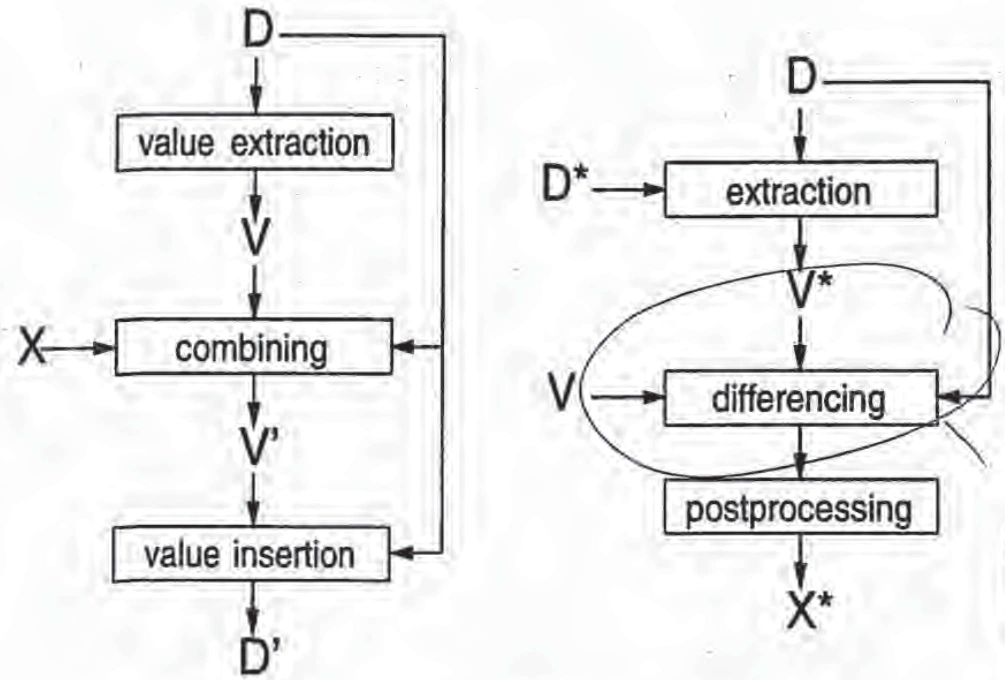


Figure 2: Immersion of the watermark in the frequency domain

structure of the watermark and its characteristics.

4 Structure of the watermark

We now give a high-level overview of our a basic watermarking scheme; many variations are possible. In its most basic implementation, a watermark consists of a sequence of real numbers $X = x_1, \ldots, x_n$. In practice, we create a watermark where each value x_i is chosen independently according to $N(0,1)$ (where $N(\mu, \sigma^2)$ denotes a normal distribution with mean μ and variance σ^2). We assume that numbers are represented by a reasonable but finite precision and ignore these insignificant roundoff errors. Section 4.1 introduces notation to describe the insertion and extraction of a watermark and Section 4.3 describes how two watermarks (the original one and the recovered, possibly corrupted one) can be compared. This procedure exploits the fact that each component of the watermark is chosen from a normal distribution. Alternative distributions are possible, including choosing x_i uniformly from $\{1, -1\}$, $\{0, 1\}$ or $[0, 1]$. However, as we discuss in Section 4.5, using such distributions leaves one particularly vulnerable to attacks using multiple watermarked documents.



differencing is used

Figure 3: Encoding and decoding of the watermark string

4.1 Description of the watermarking procedure

We extract from each document D a sequence of values $V = v_1, \ldots, v_n$, into which we insert a watermark $X = x_1, \ldots, x_n$ to obtain an adjusted sequence of values $V' = v'_1, \ldots, v'_n$. V' is then inserted back into the document in place of V to obtain a watermarked document D'. One or more attackers may then alter D', producing a new document D^*. Given D and D^*, a possibly corrupted watermark X^* is extracted and is compared to X for statistical significance. We extract X^* by first extracting a set of values $V^* = v^*_1, \ldots, v^*_n$ from D^* (using information about D) and then generating X^* from V^* and V.

Frequency-domain based methods for extracting V and V^* and inserting V' are given in Section 3. For the rest of this section we ignore the manipulations of the underlying documents.

4.2 Inserting and extracting the watermark

When we insert X into V to obtain V' we specify a scaling parameter α which determines the extent to which X alters V. Three natural formulae for computing V' are:

$$v_i' = v_i + \alpha x_i \tag{1}$$

$$v_i' = v_i(1 + \alpha x_i) \tag{2}$$

$$v_i' = v_i(e^{\alpha x_i}) \tag{3}$$

Equation 1 is always invertible, and Equations 2 and 3 are invertible if $v_i \neq 0$, which holds in all of our experiments. Given V^* we can therefore compute the inverse function to derive X^* from V^* and V.

Equation 1 may not be appropriate when the v_i values vary widely. If $v_i = 10^6$ then adding 100 may be insufficient for establishing a mark, but if $v_i = 10$ adding 100 will distort this value unacceptably. Insertion based on Equations 2 or 3 are more robust against such differences in scale. We note that Equations 2 and 3 give similar results when αx_i is small. Also, when v_i is positive then Equation 3 is equivalent to $\lg(v_i') = \lg(v_i) + \alpha x_i$, and may be viewed as an application of Equation 1 to the case where the logarithms of the original values are used.

4.2.1 Determining multiple scaling parameters

A single scaling parameter α may not be applicable for perturbing all of the values v_i, since different spectral components may exhibit more or less tolerance to modification. More generally one can have multiple scaling parameters $\alpha_1, \ldots, \alpha_n$ and use update rules such as $v_i' = v_i(1 + \alpha_i x_i)$. We can view α_i as a relative measure of how much one must alter v_i to alter the perceptual quality of the document. A large α_i means that one can perceptually "get away" with altering v_i by a large factor without degrading the document.

There remains the problem of selecting the multiple scaling values. In some cases, the choice of α_i may be based on some general assumption. For example, Equation 2 is a special case of the generalized Equation 1 ($v_i' = v_i + \alpha_i x_i$), for $\alpha_i = \alpha v_i$. Essentially, Equation 2 makes the reasonable assumption that a large value is less sensitive to additive alterations than a small value.

In general, one may have little idea of how sensitive the image is to various values. One way of empirically estimating these sensitivities is to determine the distortion caused by a number of attacks on the original image. For example, one might compute a degraded image D^* from D, extract the corresponding values $v_1^*, \ldots, v_n^*$ and choose α_i to be proportional to the deviation $|v_i^* - v_i|$. For greater robustness, one should try many forms of distortion and make α_i proportional to the average value of $|v_i^* - v_i|$. As alternatives to taking the average deviation one might also take the median or maximum deviation.

One may combine this empirical approach with general global assumptions about the sensitivity of the values. For example, one might require that $\alpha_i \geq \alpha_j$ whenever $v_i \geq v_j$. One way to combine this constraint with the empirical approach would be to set α_i according to

$$\alpha_i \sim \max_{j|v_j \leq v_i} |v_j^* - v_j|.$$

A still more sophisticated approach would be to weaken the monotonicity constraint to be robust against occasional outliers.

In all our experiments we simply use Equation 2 with a single parameter $\alpha = 0.1$. When we computed JPEG-based distortions of the original image we observed that the higher energy frequency components were not altered proportional to their magnitude (the implicit assumption of Equation 2). We suspect that we could make a less obtrusive mark of equal strength by attenuating our alterations of the high-energy components and amplifying our alterations of the lower-energy components. However, we have not yet performed this experiment.

4.3 Evaluating the similarity of watermarks

It is highly unlikely that the extracted mark X^* will be identical to the original watermark X. Even the act of requantizing the watermarked document for delivery will cause X^* to deviate from X. We measure the similarity of X and X^* by

$$\mathrm{sim}(X, X^*) = \frac{X^* \cdot X}{\sqrt{X^* \cdot X^*}}. \tag{4}$$

We argue that large values of $\mathrm{sim}(X, X^*)$ are significant by the following analysis. Suppose that the creators of document D^* had no access to X (either through the seller or through a watermarked document). Then, even conditioned on any fixed value for X^*, each x_i will be independently distributed according to $N(0,1)$. The distribution on $X^* \cdot X$ may be computed by first writing it as $\sum_{i=1}^{n} x_i^* x_i$, where x_i^* is a constant. Using the well-known formula for the distribution of a linear combination of variables that are independent and normally distributed, $X^* \cdot X$ will be distributed according to

$$N(0, \sum_{i=1}^{n} {x_i^*}^2) = N(0, X^* \cdot X^*)$$

Thus, $\mathrm{sim}(X, X^*)$ is distributed according to $N(0,1)$. We can then apply the standard significance tests for the normal distribution. For example, if X^* is created independently from X then it is extremely unlikely that $\mathrm{sim}(X, X^*) > 6$. Note that slightly higher values of $\mathrm{sim}(X, X^*)$ may be required when a large number of watermarks are on file.

4.3.1 Robust statistics

The above analysis required only the independence of X from X^*, and did not rely on any specific properties of X^* itself. This fact gives us further flexibility when it comes to preprocessing X^*. We can process X^* in a number of ways to potentially enhance our ability to extract a watermark. For example, in our experiments on images we encountered instances where the average value of x_i^*, denoted $E_i(X^*)$, differed substantially from 0, due to the effects of a dithering procedure. While this artifact could be easily eliminated as part of the extraction process, it provides a motivation for postprocessing extracted watermarks. We found that the simple transformation $x_i^* \leftarrow x_i^* - E_i(X^*)$ yielded superior values of $\mathrm{sim}(X, X^*)$. The improved performance resulted from the decreased value of $X^* \cdot X^*$; the value of $X^* \cdot X$ was only slightly affected.

In our experiments we frequently observed that x_i^* could be greatly distorted for some values of i. One postprocessing option is to simply ignore such values, setting them to 0. That is,

$$x_i^* \leftarrow \begin{cases} x_i^* & \text{if } |x_i^*| > \text{tolerance} \\ 0 & \text{Otherwise} \end{cases}$$

Again, the goal of such a transformation is to lower $X^* \cdot X^*$. A less abrupt version of this approach is to normalize the X^* values to be either $-1, 0$ or 1, by

$$x_i^* \leftarrow \mathrm{sign}(x_i^* - E_i(X^*)).$$

This transformation can have a dramatic effect on the statistical significance of the result. Other robust statistical techniques could also be used to suppress outlier effects [Hub81].

A natural question is whether such postprocessing steps run the risk of generating false positives. Indeed, the same potential risk occurs whenever there is any latitude in the procedure for extracting X^* from D^*. However, as long as the method for generating a set of values for X^* depends solely on D and D^*, our statistical significance calculation is unaffected. The only caveat to be considered is that the bound on the probability that one of $X_1^*, \ldots X_k^*$ generates a false positive is the sum of the individual bounds. Hence, to convince someone that a watermark is valid, it is necessary to have a published and rigid extraction and processing policy that is guaranteed to only generate a small number of candidate X^*.

4.4 Choosing the length, n, of the watermark

The choice of n dictates the degree to which the watermark is spread out among the relevant components of the image. In general, as the number of altered components are increased the extent to which they must be altered decreases. For a more quantitative assessment of this tradeoff, we consider watermarks of the form $v_i' = v_i + \alpha x_i$ and model a white noise attack by $v_i^* = v_i' + r_i$ where r_i are chosen according to independent normal distributions with standard deviation σ. For the watermarking procedure we described below one can recover the watermark when α is proportional to $\sigma/\sqrt{n}$. That is, by quadrupling the number of components used one can halve the magnitude of the watermark placed into each component. Note that the sum of squares of the deviations will be essentially unchanged.

However, when one increases the number of components used there is a point of diminishing returns at which the new components are randomized by trivial alterations in the image. Hence they will not be useful for storing watermark information. Thus the best choice of n is ultimately document-specific.

4.5 Resilience to multiple-document (collusion) attacks

The most general attack consists of using t multiple watermarked copies $D_1', \ldots, D_t'$ of document D to produce an unwatermarked document D^*. We note that most schemes proposed seem quite vulnerable to such attacks. As a theoretical exception, Boneh and Shaw [BS95] propose a coding scheme for use in situations in which one can insert many relatively weak 0/1 watermarks into a document. They assume that if the ith watermark is the same for all t copies of the document then it cannot be detected, changed or removed. Using their coding scheme the number of weak watermarks to be inserted scales according to t^4, which may limit its usefulness in practice.

To illustrate the power of multiple-document attacks, consider watermarking schemes in which v_i' is generated by either adding 1 or -1 at random to v_i. Then as soon as one finds two documents with unequal values for v_i' one can determine v_i and hence completely eliminate this component of the watermark. With t documents one can, on average, eliminate all but a 2^{1-t} fraction of the components of the watermark. Note that this attack does not assume anything about the distribution on v_i. While a more intelligent allocation of ± 1 values to the watermarks (following [LM93, BS95]) will better resist this simple attack, the discrete nature of the watermark components makes them much easier to completely eliminate. Our use of continuous valued watermarks appears to give greater resilience to such attacks. Interestingly, we have experimentally determined that if one chooses the x_i uniformly over some range, then one can remove the watermark using only 5 documents.

We assume an idealized scenario in which one can analyze attacks on multiple versions of a watermarked document. Let $x_{i,j}$ denote the ith component of the jth document and let $v'_{i,j} = v_i + x_{i,j}$ and $x^*_i = v^*_i - v_i$. We assume that $x_{i,j}$ is independently distributed according to $N(0,1)$ and that v_i is independently distributed and has a locally "flat" probability distribution. What we mean by flat is made clearer in the following analysis.

Given $v'_{i,1}, \ldots, v'_{i,t}$, let $\bar{v}_i = \frac{1}{t}\sum_j v'_{i,j}$. Assuming that the prior distribution of v_i is essentially constant around $\bar{v}$ (our flatness assumption) then the posterior distribution on v_i is essentially equal to $N(\bar{v}, 1/t)$. Note that this distribution depends only on $\bar{v}$ and has no other relationship to $v'_{i,1}, \ldots, v'_{i,t}$.

We argue that regardless of the strategy used for generating V^*, there is some j such that the expected value of $X^* \cdot X_j$ is $\Omega(n/t)$, where $X_j = x_{1,j}, \ldots, x_{n,j}$. Here the expectation is given over the posterior distribution on V. It suffices to prove that

$$E\left(\sum_{j=1}^{t} X^* \cdot X_j\right) = \Omega(n),$$

and hence that

$$E\left(\sum_{j=1}^{t} x^*_i x_{i,j}\right) = \Omega(1) \tag{5}$$

for all i.

For notational convenience we omit the i in the subscripts in the remaining analysis. Let $v = \bar{v} + \delta$, $v^* = \bar{v} + \delta^*$ and $v'_{i,j} = \bar{v} + \delta'_j$. Thus, δ is distributed according to $N(0, 1/t)$ and $\sum_j \delta'_j = 0$. Let $\rho_\delta(x)$ denote the density function for δ. We can express the left side of Equation 5 by the integral,

$$\int_{-\infty}^{\infty} \rho_\delta(x)dx \sum_{j=1}^{t}(\delta^* - \delta)(\delta'_j - \delta) \tag{6}$$

Having x range over $[-\infty, \infty]$ seems problematic, since the flatness approximation would certainly be violated over this range. However, the contribution to the integral comes almost entirely from the region where x is close to 0, since the $\rho_\delta(x)$ term is inversely exponential in x^2, so the approximation is indeed reasonable. By straightforward manipulations, Equation 6 can be expressed as

$$\int_{-\infty}^{\infty} \rho_\delta(x)dx \sum_{j=1}^{t}(\delta^* - \delta)\delta'_j + t\int_{-\infty}^{\infty} \rho_\delta(x)dx\delta^2 - t\delta^* \int_{-\infty}^{\infty} \rho_\delta(x)dx\delta. \tag{7}$$

Figure 4: "Bavarian Couple" courtesy of Corel Stock Photo Library.

First, we observe that

$$\sum_{j=1}^{t}(\delta^{*}-\delta)\delta'_j = 0,$$

since $\sum_j \delta'_j = 0$ and hence the first term of Equation 7 vanishes. Next, by the properties of the normal distribution,

$$\begin{aligned} \int_{-\infty}^{\infty} p_{\delta}(x)dx\delta^2 &= 1/t \text{ and,} \\ \int_{-\infty}^{\infty} p_{\delta}(x)dx\delta &= 0 \end{aligned}$$

Thus Equation 5 is identically equal to 1.

5 Experimental Results

In order to evaluate the proposed digital watermark, we first took the "Bavarian Couple"[2] image of Figure (4) and produced the watermarked version of Figure (5)

[2]The common test image "Lenna" was originally used in our experiments and similar results were obtained. However, questions of taste aside, Playboy Inc. refused to grant copyright permission for electronic distribution.

Figure 5: Watermarked version of "Bavarian Couple".

5.1 Experiment 1: Uniqueness of watermark

Figure (6) shows the response of the watermark detector to 1000 randomly generated watermarks of which only one matches the watermark present in Figure (5). The positive response due to the correct watermark is very much stronger that the response to incorrect watermarks, suggesting that the algorithm has very low false positives (and false negative) response rates.

5.2 Experiment 2: Image Scaling

The watermarked image was scaled to half its orginal size, Figure (7a). In order to recover the watermark, the quarter-sized image was re-scaled to its original dimensions, as shown in Figure (7b), in which it is clear that considerable fine detail has been lost in the scaling process. This is to be expected since subsampling of the image requires a low pass spatial filtering operation. The response of the watermark detector to the original watermarked image of Figure (5) was 32.0 which compares to a response of 13.4 for the re-scaled version of Figure (7b). While the detector response is down by over 50%, the response is still well above random chance levels suggesting that the watermark is robust to geometric distortions. Moreover, it should be noted that 75% of the original data is missing from the scaled down image of Figure 7.

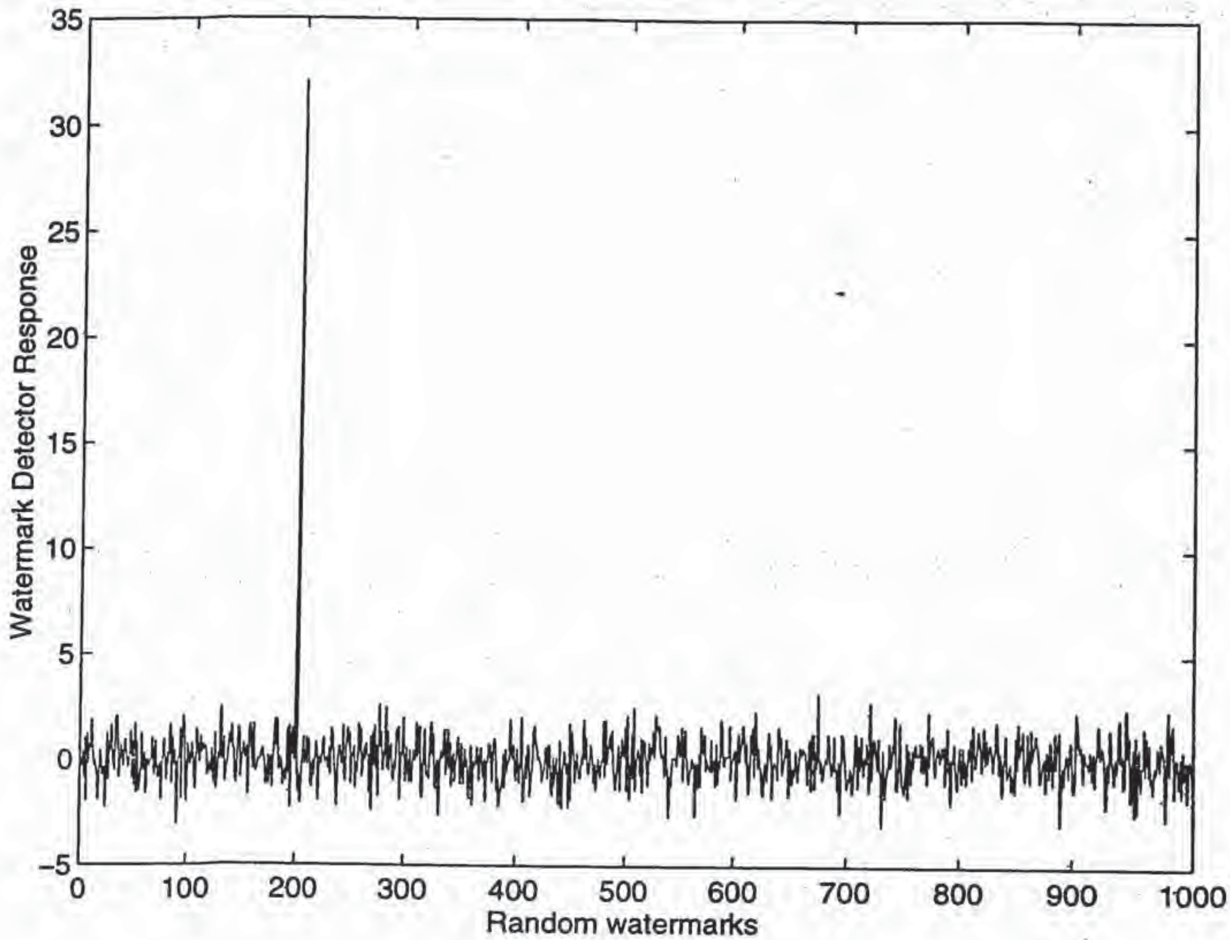


Figure 6: Watermark detector response to 1000 randomly generated watermarks. Only one watermark (the one to which the detector was set to respond) matches that present in Figure (5).

Figure 7: (a) Low pass filtered, 0.5 scaled image of "Bavarian Couple", (b) re-scaled image showing noticable loss of fine detail.

5.3 Experiment 3: JPEG coding distortion

Figure (8) shows a JPEG encoded version of "Bavarian Couple" with parameters of 10% quality and 0% smoothing, which results in clearly visible distortions of the image. The response of the watermark detector is 22.8, again suggesting that the algorithm is robust to common encoding distortions. Figure (9) shows a JPEG encoded version of "Bavarian Couple" with parameters of 5% quality and 0% smoothing, which results is very significant distortions of the image. The response of the watermark detector in this case is 13.9, which is still well above random.

5.4 Experiment 4: Dithering Distortion

Figure (10) shows a dithered version of "Bavarian Couple". The response of the watermark detector is 5.2 again suggesting that the algorithm is robust to common encoding distortions. In fact, more reliable detection can be achieved simply by removing any non-zero mean from the extracted watermark, as discussed in Section 4.3.1. In this case the detection value is 10.5.

Figure 8: JPEG encoded version of "Bavarian Couple" with 10% quality and 0% smoothing.

Figure 9: JPEG encoded version of "Bavarian Couple" with 5% quality and 0% smoothing.

Figure 10: Dithered version of "Bavarian Couple".

5.5 Experiment 5: Clipping

Figure (11a) shows a clipped version of the watermarked image of Figure (5) in which only the central quarter of the image remains. In order to extract the watermark from this image, the missing portions of the image were replaced with portions from the original **unwatermarked** image of Figure (4), as shown in Figure (11b). In this case, the response of the watermark is 14.6. Once again, this is well above random even though 75% of the data has been removed.

Figure (12a) shows a clipped version of the JPEG encoded image of Figure (8) in which only the central quarter of the image remains. As before, the missing portions of the image were replaced with portions from the original **unwatermarked** image of Figure (4), as shown in Figure (12b). In this case, the response of the watermark is 10.6. Once more, this is well above random even though 75% of the data has been removed and distortion is present in the clipped portion of the image.

5.6 Experiment 6: Print, xerox and scan

Figure (13) shows an image of Lenna after (1) printing, (2) xeroxing, then (3) scanning at 300 dpi using UMAX PS-2400X scanner, and finally (4) rescaled to a size of 256×256. Clearly, this image suffers from several levels of distortion that accompany each of the four stages. High frequency pattern noise is especially

Figure 11: (a) Clipped version of watermarked "Bavarian Couple", (b) Restored version of "Bavarian Couple" in which missing portions have been replaced with imagery [illegible] Figure (4).

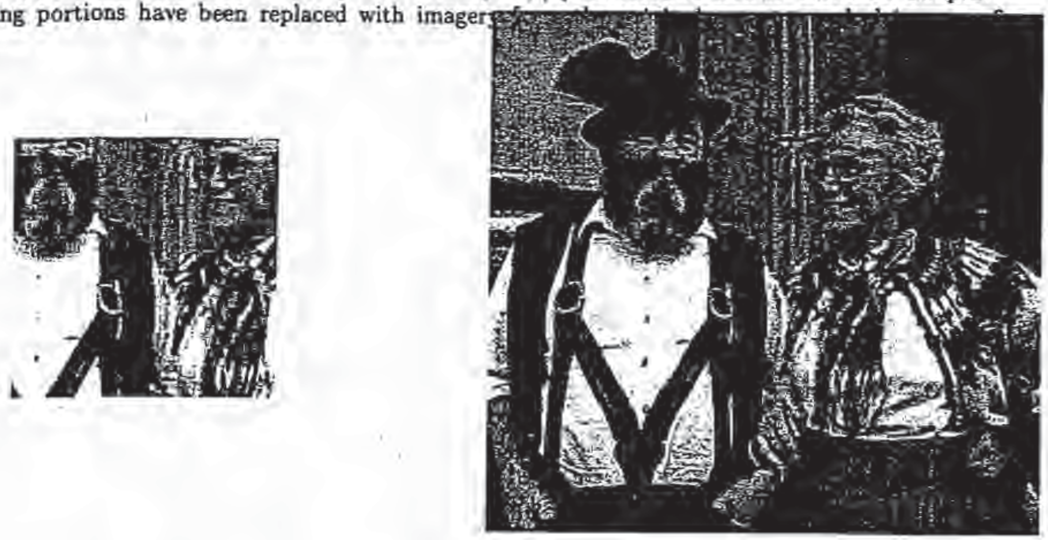


Figure 12: (a) Clipped version of JPEG encoded (10% quality, 0% smoothing) "Bavarian Couple", (b) Restored version of "Bavarian Couple" in which missing portions have been replaced with imagery from the original unwatermarked image of Figure (4).

Figure 13: Printed, xeroxed, scanned and rescaled image of "Bavarian Couple".

noticeable. The detector response to the watermark is 4.0. However, if the non-zero mean is removed and only the sign of the elements of the watermark are used, then the detector response is 7.0, which is well above random.

5.7 Experiment 7: Attack by watermarking watermarked images

Figure (14) shows an image of "Bavarian Couple" after five successive watermarking operations, i.e. the original image is watermarked, the watermarked image is watermarked, etc. This may be considered another form of attack in which it is clear that significant image degradation eventually occurs as the process is repeated. This attack is equivalent to adding noise to the frequency bins containing the watermark. Interestingly, Figure (15) shows the response of the detector to 1000 randomly generated watermarks, which include the five watermarks present in the image. Five spikes clearly indicate the presence of the five watermarks and demonstrate that successive watermarking does not interfere with the process.

5.8 Experiment 8: Attack by collusion

In a similar experiment, we took five separately watermarked images and averaged them to form Figure (16) in order to simulate a simple collusion attack. As before, Figure (17) shows the response of the detector to 1000 randomly generated watermarks, which include the five watermarks present in the image. Once again,

Figure 14: Image of "Bavarian Couple" after five successive watermarks have been added.

five spikes clearly indicate the presence of the five watermarks and demonstrate that simple collusion based on averaging a few images is ineffective.

6 Conclusion

A need for electronic watermarking is developing as electronic distribution of copyright material becomes more prevalent. Above, we outlined the necessary characteristics of such a watermark. These are: fidelity preservation, robustness to common signal and geometric processing operations, robustness to attack, and applicability to audio, image and video data.

To meet these requirements, we proposed a watermark whose structure consisted of 1000 randomly generated numbers with a Normal distribution having zero mean and unity variance. A binary watermark was rejected based on the fact that it is much less robust to attacks based on collusion of several independently watermarked copies of an image. The length of the watermark is variable and can be adjusted to suit the characteristics of the data. For example, longer watermarks might be used for an image that is especially sensitive to large modifications of its spectral coefficients, thus requiring weaker scaling factors for individual components.

The watermark is then placed in the perceptually *most* significant components of the image spectrum.

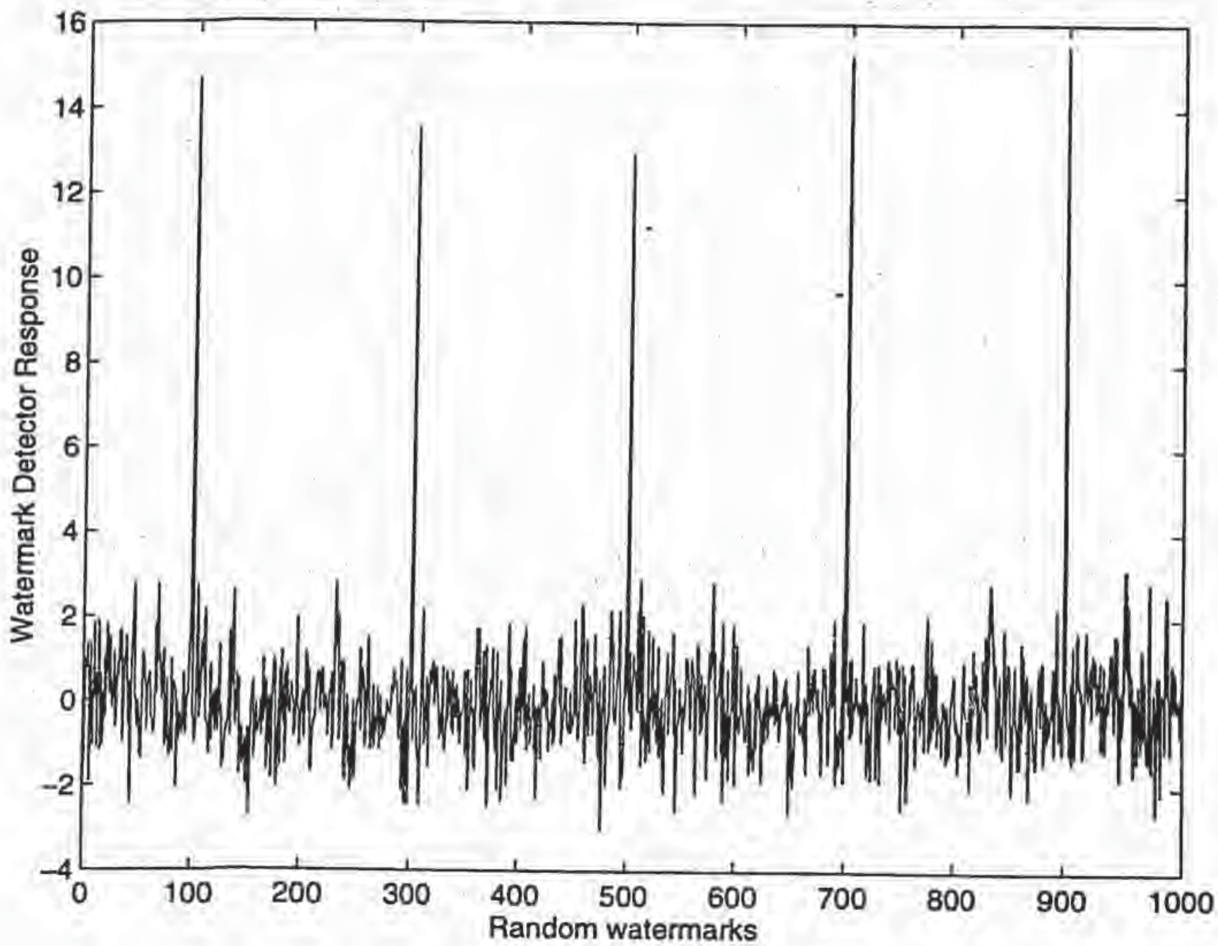


Figure 15: Watermark detector response to 1000 randomly generated watermarks (including the 5 specific watermarks) for the watermarked image of Figure (14). Each of the five watermarks is clearly indicated.

Figure 16: Image of "Bavarian Couple" after averaging together five independently watermarks versions of the "Bavarian Couple" image.

This ensures that the watermark remains with the image even after common signal and geometric distortions. Modification of these spectral components results in severe image degradation long before the watermark itself is destroyed. Of course, to insert the watermark, it is necessary to alter these very same coefficients. However, each modification can be extremely small and, in a manner similar to spread spectrum communication, a strong narrowband watermark may be distributed over a much broader image (channel) spectrum. Conceptually, detection of the watermark then proceeds by adding all of these very small signals, and concentrating them once more into a signal with high signal-to-noise ratio. Because the magnitude of the watermark at each location is only known to the copyright holder, an attacker would have to add much more noise energy to each spectral coefficient in order to be sufficiently confident of removing the watermark. However, this process would destroy the image.

In our experiments, we added the watermark to the image by modifying 1000 of the more perceptually significant components of the image spectrum. More specifically, the 1000 largest coefficients of the DCT (excluding the DC term) were used. Further refinement of the method would identify perceptually significant components based on an analysis of the image and the human perceptual system and might also include additional considerations regarding the relative predictability of a frequency based on its neighbors. The latter property is important to consider in order to minimize any attack based on a statistical analysis of

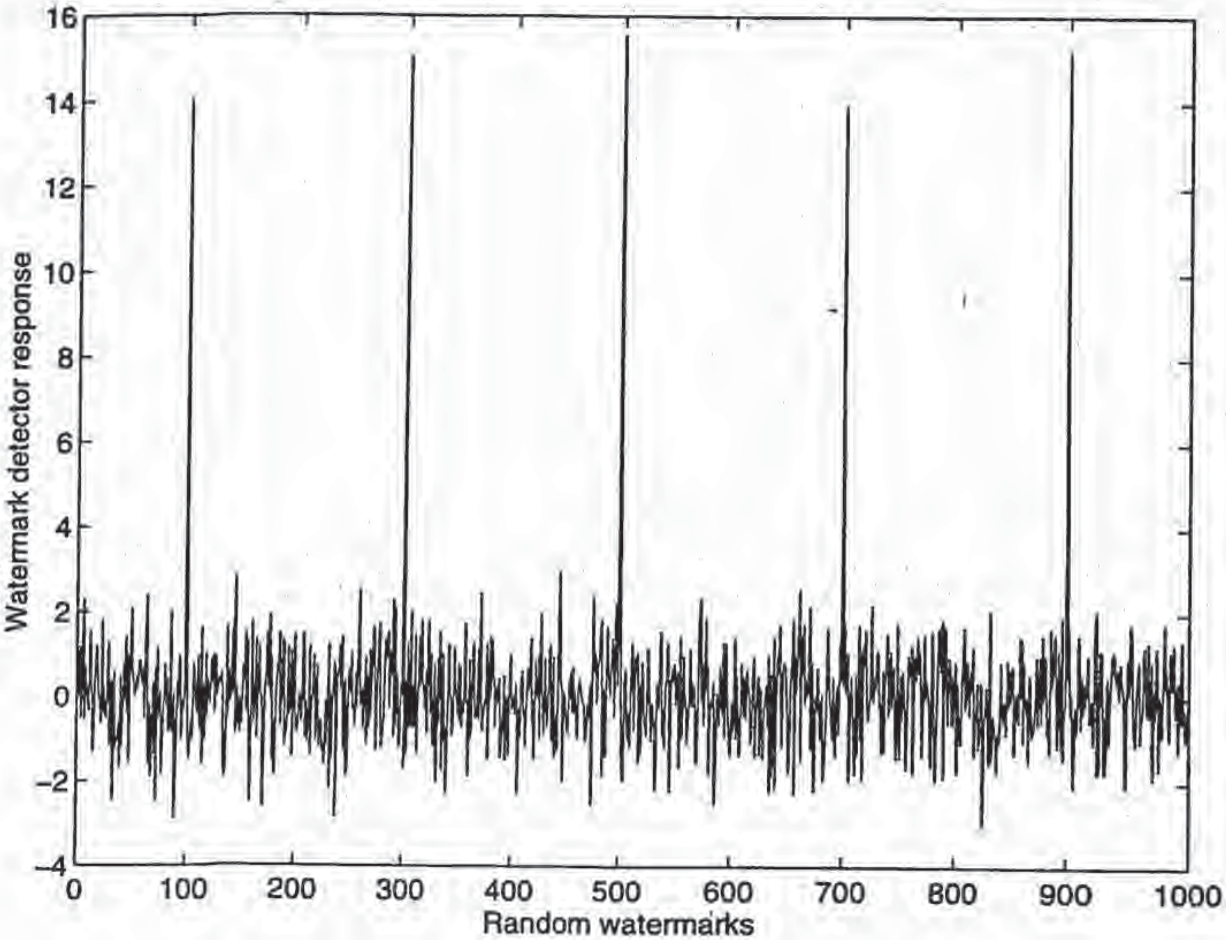


Figure 17: Watermark detector response to 1000 randomly generated watermarks (including the 5 specific watermarks) for the watermarked image of Figure (16). Each of the five watermarks is clearly detected, indicating that collusion by averaging is ineffective.

frequency spectra that attempts to replace components with their maximul likelihood estimate, for example. The choice of the DCT is not critical to the algorithm ~~and other spectral~~ transforms, including wavelet type decompositions are also possible. In fact, use of the FFT rather than DCT may prefereble from a computational perspective.

It was shown, using the "Lenna" image, that the algorithm can extract a reliable copy of the watermark from imagery that has been significantly degraded through several common geometric and signal processing procedures. These include, zooming (low pass filtering), cropping, lossy JPEG encoding, dithering, printing, photocopying and subsequent rescanning.

More experimental work needs to be performed to validate these results over a wide class of data. Application of the method to color images should be straightforward though robustness to certain color image processing procedures should be investigated. Similarly, the system should work well on text images, however, the binary nature of the image together with its much more structured spectral distribution need more work. Furthermore, application of the watermarking method to audio and video data should follow in a straightforward fashion, although, attention must be paid to the time varying nature of these data. A more sophisticated watermark verification process may also be ~~possible using methods developed~~ for spread spectrum communications.

Larger system issues must be also addressed in order for this system to be used in practice. For example, it would be useful to be able to prove in court that a watermark is present without publically revealing the original, unmarked document. This is not hard to accomplish using secure trusted hardware; an efficient purely cryptographic solution seems much more difficult. It should also be noted that current proposal only allows the watermark to be extracted by the owner, since the original unwatermarked image is needed as part of the extraction process. This prohibits potential users from querying the image for ownership and copyright information. This capability may be desirable but appears difficult to achieve with the same level of robustness. However, it is straightforward to provide if a much weaker level of protection is acceptable and might therefore be added as a secondary watermarking procedure. Finally, we note that while the proposed methodology is used to hide watermarks in data, the same process can be applied to sending other forms of message through media data.

[KRZ94] E. Koch, J. Rindfrey, and J. Zhao. Copyright protection for multimedia data. In *Proc. of the Int. Conf. on Digital Media and Electronic Publishing*, 1994.

[KZ95] E. Koch and Z. Zhao. Towards robust and hidden image copyright labeling. In *Proceedings of 1995 IEEE Workshop on Nonlinear Signal and Image Processing*, June 1995.

[Lim90] J.S Lim. *Two-Dimensional Signal Processing*. Prentice Hall, Englewood Cliffs, N.J., 1990.

[LM93] F. T. Leighton and S. Micali. Secret-key agreement without public-key cryptography. In *Proceedings of Crypto*, 1993.

[MQ95] B. M. Macq and J-J Quisquater. Cryptology for digital tv broadcasting. *Proc. of the IEEE*, 83(6):944–957, 1995.

[MT94] K. Matsui and K. Tanaka. Video-steganography. In *IMA Intellectual Property Project Proceedings*, volume 1, pages 187–206, 1994.

[PSM82] R. L. Pickholtz, D. L. Schilling, and L. B. Millstein. Theory of spread spectrum communications - a tutorial. *IEEE Trans. on Communications*, pages 855–884, 1982.

[SLAN91] W. F. Schreiber, A. E. Lippman, E. H. Adelson, and A. N. Netravali. Receiver-compatible enhanced definition television system. Technical Report 5,010,405, United States Patent, 1991.

[TNM90] K. Tanaka, Y. Nakamura, and K. Matsui. Embedding secret information into a dithered multi-level image. In *Proc, 1990 IEEE Military Communications Conference*, pages 216–220, 1990.

[Tur89] L. F. Turner. Digital data security system. Patent IPN WO 89/08915, 1989.

[vSTO94] R. G. van Schyndel, A. Z. Tirkel, and C. F. Osborne. A digital watermark. In *Int. Conf. on Image Processing*, volume 2, pages 86–90. IEEE, 1994.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☑ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

colleagues on *Newsday*,
taught me most of what
Bernie Bookbinder, who
lways be paramount; to
cend itself; and to Stan
at the time but has since

cares to tell me of any
iscences. I shall be very

DAVID KAHN

THE CODE BREAKERS 22

A FEW WORDS

EVERY TRADE has its vocabulary. That of cryptology is simple, but even so a familiarity with its terms facilitates understanding. A glossary may also serve as a handy reference. The definitions in this one are informal and ostensive. Exceptions are ignored and the host of minor terms are not defined—the text covers these when they come up.

The **plaintext** is the message that will be put into secret form. Usually the plaintext is in the native tongue of the communicators. The message may be hidden in two basic ways. The methods of **steganography** conceal the very existence of the message. Among them are invisible inks and microdots and arrangements in which, for example, the first letter of each word in an apparently innocuous text spells out the real message. (When steganography is applied to electrical communications, such as a method that transmits a long radio message in a single short spurt, it is called **transmission security**.) The methods of **cryptography**, on the other hand, do not conceal the presence of a secret message but render it unintelligible to outsiders by various transformations of the plaintext.

Two basic transformations exist. In **transposition**, the letters of the plaintext are jumbled; their normal order is disarranged. To shuffle *secret* into ETCRSE is a transposition. In **substitution**, the letters of the plaintext are replaced by other letters, or by numbers or symbols. Thus *secret* might become 19 5 3 18 5 20, or XIWOXV in a more complicated system. In transposition, the letters retain their identities—the two *e*'s of *secret* are still present in ETCRSE—but they lose their positions, while in substitution the letters retain their positions but lose their identities. Transposition and substitution may be combined.

Substitution systems are much more diverse and important than transposition systems. They rest on the concept of the **cipher alphabet**. This is the list of equivalents used to transform the plaintext into the secret form. A sample cipher alphabet might be:

plaintext letters	a b c d e f g h i j k l m n o p q r s t u v w x y z
cipher letters	L B Q A C S R D T O F V M H W I J X G K Y U N Z E P

This graphically indicates that the letters of the plaintext are to be replaced

A FEW WORDS

EVERY TRADE has its vocabulary. That of cryptology is simple, but even so a familiarity with its terms facilitates understanding. A glossary may also serve as a handy reference. The definitions in this one are informal and ostensive. Exceptions are ignored and the host of minor terms are not defined—the text covers these when they come up.

The plaintext is the message that will be put into secret form. Usually the plaintext is in the native tongue of the communicators. The message may be hidden in two basic ways. The methods of **steganography** conceal the very existence of the message. Among them are invisible inks and microdots and arrangements in which, for example, the first letter of each word in an apparently innocuous text spells out the real message. (When steganography is applied to electrical communications, such as a method that transmits a long radio message in a single short spurt, it is called **transmission security**.) The methods of **cryptography**, on the other hand, do not conceal the presence of a secret message but render it unintelligible to outsiders by various transformations of the plaintext.

Two basic transformations exist. In **transposition**, the letters of the plaintext are jumbled; their normal order is disarranged. To shuffle *secret* into ETCRSE is a transposition. In **substitution**, the letters of the plaintext are replaced by other letters, or by numbers or symbols. Thus *secret* might become 19 5 3 18 5 20, or XIWOXV in a more complicated system. In transposition, the letters retain their identities—the two *e*'s of *secret* are still present in ETCRSE—but they lose their positions, while in substitution the letters retain their positions but lose their identities. Transposition and substitution may be combined.

Substitution systems are much more diverse and important than transposition systems. They rest on the concept of the **cipher alphabet**. This is the list of equivalents used to transform the plaintext into the secret form. A sample cipher alphabet might be:

plaintext letters	a b c d e f g h i j k l m n o p q r s t u v w x y z
cipher letters	L B Q A C S R D T O F V M H W I J X G K Y U N Z E P

This graphically indicates that the letters of the plaintext are to be replaced

This is the only mention of writing in the *Iliad.* Homer's language is not precise enough to tell exactly what the markings on the tablets were. They were probably nothing more than ordinary letters—actual substitution of symbols for letters seems too sophisticated for the era of the Trojan War. But the mystery that Homer throws around the tablets does suggest that some rudimentary form of concealment was used, perhaps some such allusion as "Treat this man as well as you did Glaucus," naming someone whom the king had had assassinated. The whole tone of the reference makes it fairly certain that here, in the first great literary work of European culture, appear that culture's first faint glimmerings of secrecy in communication.

A few centuries later, those glimmerings had become definite beams of light. Several stories in the *Histories* of Herodotus deal specifically with methods of steganography (not, however, with cryptography). Herodotus tells how a Median noble named Harpagus wanted to avenge himself on his relative, the king of the Medes, who years before had tricked him into eating his own son. So he hid a message to a potential ally in the belly of an unskinned hare, disguised a messenger as a hunter, and sent him off down the road, carrying the hare as if he had just caught it. The road guards suspected nothing, and the messenger reached his destination. At it was Cyrus, king of Persia, whose country was then subject to Medea and who had himself been the target of a babyhood assassination attempt by the Medean king. The message told him that Harpagus would work from within to help him dethrone the Medean king. Cyrus needed no further urging. He led the Persians in revolt; they defeated the Medes and captured the king, and Cyrus was on his way to winning the epithet "the Great."

Herodotus tells how another revolt—this one against the Persians—was set in motion by one of the most bizarre means of secret communication ever recorded. One Histiaeus, wanting to send word from the Persian court to his son-in-law, the tyrant Aristagoras at Miletus, shaved the head of a trusted slave, tattooed the secret message thereon, waited for a new head of hair to grow, then sent him off to his son-in-law with the instruction to shave the slave's head. When Aristagoras had done so, he read on the slave's scalp the message that urged him to revolt against Persia.

One of the most important messages in the history of Western civilization was transmitted secretly. It gave to the Greeks the crucial information that Persia was planning to conquer them. According to Herodotus,

> The way they received the news was very remarkable. Demaratus, the son of Ariston, who was an exile in Persia, was not, I imagine—and as is only natural to suppose—well disposed toward the Spartans; so it is open to question whether what he did was inspired by benevolence or malicious pleasure. Anyway, as soon as news reached him at Susa that Xerxes had decided upon the invasion of Greece, he felt that he must pass on the information to Sparta. As the danger of discovery was great, there was only one way in which he could contrive to get the message through: this was by scraping the wax off a pair of wooden folding

> tablets, writing on the wood underneath what Xerxes intended to do, and then covering the message over with wax again. In this way the tablets, being apparently blank, would cause no trouble with the guards along the road. When the message reached its destination, no one was able to guess the secret until, as I understand, Cleomenes' daughter Gorgo, who was the wife of Leonidas, discovered it and told the others that, if they scraped the wax off, they would find something written on the wood underneath. This was done; the message was revealed and read, and afterwards passed on to the other Greeks.

The rest is well-known. Thermopylae, Salamis, and Plataea ended the danger that the flame of Western civilization would be extinguished by an Oriental invasion. The story is not without a certain bitter irony, however, for Gorgo, who may be considered the first woman cryptanalyst, in a way pronounced a death sentence on her own husband: Leonidas died at the head of the heroic band of Spartans who held off the Persians for three crucial days at the narrow pass of Thermopylae.

It was the Spartans, the most warlike of the Greeks, who established the first system of military cryptography. As early as the fifth century B.C., they employed a device called the "skytale," the earliest apparatus used in cryptology and one of the few ever devised in the whole history of the science for transposition ciphers. The skytale consists of a staff of wood around which a strip of papyrus or leather or parchment is wrapped close-packed. The secret message is written on the parchment down the length of the staff; the parchment is then unwound and sent on its way. The disconnected letters make no sense unless the parchment is rewrapped around a baton of the same thickness as the first: then words leap from loop to loop, forming the message.

Thucydides tells how it enciphered a message from the ephors, or rulers, of Sparta, ordering the too-ambitious Spartan prince and general Pausanius to follow the herald back home from where he was trying to ally himself with the Persians, or have war declared against him by the Spartans. He went. That was about 475 B.C. About a century later, according to Plutarch, another skytale message recalled another Spartan general, Lysander, to face charges of insubordination. Xenophon also records the skytale's use in enciphering a list of names in an order sent to another Spartan commander.

The world owes its first instructional text on communications security to the Greeks. It appeared as an entire chapter in one of the earliest works on military science, *On the Defense of Fortified Places*, by Aeneas the Tactician. He retold some of Herodotus' stories, and listed several systems. One replaced the vowels of the plaintext by dots—one dot for alpha, two for epsilon, and so on to seven for omega. Consonants remained unenciphered. In a steganographic system, holes representing the letters of the Greek alphabet were bored through an astragal or a disk. Then the encipherer passed yarn through the holes that successively represented the letters of his message. The decipherer would presumably have to reverse the entire text after unraveling the thread. Another steganographic system was still in use in the 20th century:

Aeneas suggested pricking holes in a book or other document above or below the letters of the secret message. German spies used this very system in World War I, and used it with a slight modification in World War II—dotting the letters of newspapers with invisible ink.

Another Greek writer, Polybius, devised a system of signaling that has been adopted very widely as a cryptographic method. He arranged the letters in a square and numbered the rows and columns. To use the English alphabet, and merging *i* and *j* in a single cell to fit the alphabet into a 5 × 5 square:

	1	2	3	4	5
1	a	b	c	d	e
2	f	g	h	ij	k
3	l	m	n	o	p
4	q	r	s	t	u
5	v	w	x	y	z

Each letter may now be represented by two numbers—that of its row and that of its column. Thus *e* = 15, *v* = 51. Polybius suggested that these numbers be transmitted by means of torches—one torch in the right hand and five in the left standing for *e*, for example. This method could signal messages over long distances. But modern cryptographers have found several characteristics of the Polybius square, or "checkerboard," as it is now commonly called, exceedingly valuable—namely, the conversion of letters to numbers, the reduction in the number of different characters, and the division of a unit into two separately manipulable parts. Polybius' checkerboard has therefore become very widely used as the basis of a number of systems of encipherment.

These Greek authors never said whether any of the substitution ciphers they described were actually used, and so the first attested use of that genre in military affairs come from the Romans—and from the greatest Roman of them all, in fact. Julius Caesar tells the story himself in his *Gallic Wars*. He had proceeded by forced marches to the borders of the Nervii, and

> There he learned from prisoners what was taking place at Cicero's station, and how dangerous was his case. Then he persuaded one of the Gallic troopers with great rewards to deliver a letter to Cicero. The letter he sent written in Greek characters, lest by intercepting it the enemy might get to know of our designs. The messenger was instructed, if he could not approach, to hurl a spear, with the letter fastened to the thong, inside the entrenchment of the camp. In the dispatch he wrote that he had started with the legions and would speedily be with him, and he exhorted Cicero to maintain his old courage. Fearing danger, the Gaul discharged the spear, as he had been instructed. By chance it stuck fast in the tower, and for two days was not sighted by our troops; on the third day it was sighted by a soldier, taken down, and delivered to Cicero. He read it through and then recited it at a parade of the troops, bringing the greatest rejoicing to all.

16

CENSORS, SCRAMBLERS, AND SPIES

CIPHER IS THE LANGUAGE OF SPIES—and usually they must talk in whispers. A spy's success, his very existence, depends on his not being seen or heard. Sending messages in obviously cryptographic form would alert counterespionage to him as effectively as wearing a cloak and dagger. Yet he must transmit, else he is useless. So he eschews the overt methods of secret communications for the covert. He resorts to open codes, hollow heels, invisible inks, microscopically small missives—the steganographic methods that conceal the very fact that a message is being sent. He seeks to communicate unnoticed.

And to block this very attempt and root out the enemy within, governments erect great filters at their mail and cable ports of entry to prevent and detect these clandestine communications. These sieves, which let innocent messages flow through, are the censorship organizations.

Descended in a sense from the black chambers of the 1700s, they are creatures of war in democracies and of tyranny in dictatorships. Censorship first sprang up on a major scale in World War I, and the lessons that Britain learned then she put to good use twenty years later when she again filtered communications. Even before the United States entered the war, British censorship had caught two major German spies in the United States and its protectorate of Cuba.

In December, 1940, one of the 1,200 examiners that British censorship had installed in the commodious Princess Hotel in Bermuda stopped a letter addressed to Berlin from New York. He suspected it because it described a list of Allied shipping and used several expressions—such as "cannon" for "guns" in describing the vessels' armament—that suggested the writer might be German and a possible Nazi agent. The letter was signed "Joe K." A watch set up for more letters with his handwriting soon picked out quite a few more, mostly to Spain and Portugal. Their language seemed slightly forced, and a team began studying the letters to see whether this indicated an open code and, if so, what the real meaning was.

One member of the team was a persistent young woman named Nadya Gardner, who became convinced that the letters contained invisible writing. The usual strip tests with chemicals that bring out the ordinary secret inks

nally the chemists, under
:nted back in World War
in the back of the typed
o Mr. Manuel Alonso,
pages a list of ships then
anhattan) the Norwegian
90 was a Dutch freighter
abel Machado Santos in
e about 70,000 men on
pril 14—many thanks
m letter 69)—3. Boeing
S. Army to Britain on
a solution of pyramidon,
dily obtainable at most

etters bore no return ad-
the spy's real first name
ut another Joe K letter
in a New York traffic
Hospital. F.B.I. agents
io Lopez Lido, and that
d his briefcase after the
earned that Lido's true
of the Joe K letters was
raised in Germany, had
anize a spy ring, which

on in his possession. The
inted for by its double
my limited facilities and
r completely," he wrote
y, cover addresses for
have difficulty fulfilling
cause of too few agents
U.S. District Court at

ation went to his death.
a rather Germanic cast
Havana to Lisbon and
on was confirmed when
in Havana harbor and
rs were alerted to watch
few days later. Censor-
ed details of merchant

The Code Breakers

shipping in Cuban waters and of the enlargement of the U.S. Navy's base at Guantánamo Bay, until the writer's real Havana address showed up in secret ink. Letters posted to this address were watched, and on September 5, 1942, after sufficient evidence had been amassed, police arrested "R. Castillo," who proved to be Heinz August Luning. He had been sent to Havana from Germany in September, 1941, and of the 48 letters he had sent to Europe, the Bermuda censors had intercepted all but five. On November 9, 1942, he went before a firing squad at Principe Fortress, the first man in Cuba to be executed as a spy.

Soon after Pearl Harbor, the United States built up a censorship service that began in the borrowed office in which Byron Price went to work as Chief Censor and grew to an organization whose 14,462 examiners occupied 90 buildings throughout the country, opened a million pieces of overseas mail a day, listened to innumerable telephone conversations, and scanned movies, magazines, and radio scripts. Millions became familiar with the "Opened by Censor" sticker and the scissored letter.

To plug up as many steganographic channels of communication as possible, the Office of Censorship banned in advance the sending of whole classes of objects or kinds of messages. International chess games by mail were stopped. Crossword puzzles were extracted from letters, for the examiners did not have time to solve them to see if they concealed a secret message, and so were newspaper clippings, which might have spelled out messages by dotting successive letters with secret ink—a modern version of a system described more than 2,000 years earlier by Aeneas the Tactician. Listing of students' grades was tabooed. One letter containing knitting instructions was held up long enough for an examiner to knit a sweater to see if the given sequence of knit two and cast off contained a hidden message like that of Madame Defarge, who knitted into her "shrouds" the names of further enemies of the French Republic, "whose lives the guillotine then surely swallowed up." A stamp bank was maintained at each censorship station; examiners removed loose stamps, which might spell out a code message, and replaced them with others of equal value, but of different number and denomination. Blank paper, often sent from the United States to relatives in paper-short countries, was similarly replaced from a paper bank to obviate secret-ink transmissions. Childish scrawls, sent from proud parents to proud grandparents, were removed because of the possibility of their covering a map. Even lovers' X's, meant as kisses, were heartlessly deleted if censors thought they might be a code.

Censorship cable regulations prohibited sending any text that was unclear to the censor, including numbers unrelated to the text or a personal note in a business communication, and that was not in English, French, Spanish, or Portuguese plain language. To kill any possible sub rosa message, censors sometimes paraphrased messages. This practice gave rise to Censorship's classic tale, which dates back to World War I. Onto the desk of a censor

nally the chemists, under
nted back in World War
n the back of the typed
, Mr. Manuel Alonso,
pages a list of ships then
inhattan: the Norwegian
90) was a Dutch freighter
ibel Machado Santos in
about 70,000 men on
pril 14—many thanks #
n letter 69)—3. Boeing
S. Army to Britain on
solution of pyramidon,
ily obtainable at most

tters bore no return ad-
he spy's real first name
at another Joe K letter
in a New York traffic
Hospital. F.B.I. agents
o Lopez Lido, and that
d his briefcase after the
arned that Lido's true
f the Joe K letters was
aised in Germany, had
nize a spy ring, which

n in his possession. The
nted for by its double
ny limited facilities and
completely," he wrote
. cover addresses for
ave difficulty fulfilling
cause of too few agents
U.S. District Court at

tion went to his death.
a rather Germanic cast
Havana to Lisbon and
n was confirmed when
n Havana harbor and
s were alerted to watch
ew days later. Censor-
d details of merchant

shipping in Cuban waters and of the enlargement of the U.S. Navy's base at Guantánamo Bay, until the writer's real Havana address showed up in secret ink. Letters posted to this address were watched, and on September 5, 1942, after sufficient evidence had been amassed, police arrested "R. Castillo," who proved to be Heinz August Luning. He had been sent to Havana from Germany in September, 1941, and of the 48 letters he had sent to Europe, the Bermuda censors had intercepted all but five. On November 9, 1942, he went before a firing squad at Principe Fortress, the first man in Cuba to be executed as a spy.

Soon after Pearl Harbor, the United States built up a censorship service that began in the borrowed office in which Byron Price went to work as Chief Censor and grew to an organization whose 14,462 examiners occupied 90 buildings throughout the country, opened a million pieces of overseas mail a day, listened to innumerable telephone conversations, and scanned movies, magazines, and radio scripts. Millions became familiar with the "Opened by Censor" sticker and the scissored letter.

To plug up as many steganographic channels of communication as possible, the Office of Censorship banned in advance the sending of whole classes of objects or kinds of messages. International chess games by mail were stopped. Crossword puzzles were extracted from letters, for the examiners did not have time to solve them to see if they concealed a secret message, and so were newspaper clippings, which might have spelled out messages by dotting successive letters with secret ink—a modern version of a system described more than 2,000 years earlier by Aeneas the Tactician. Listing of students' grades was tabooed. One letter containing knitting instructions was held up long enough for an examiner to knit a sweater to see if the given sequence of knit two and cast off contained a hidden message like that of Madame Defarge, who knitted into her "shrouds" the names of further enemies of the French Republic, "whose lives the guillotine then surely swallowed up." A stamp bank was maintained at each censorship station; examiners removed loose stamps, which might spell out a code message, and replaced them with others of equal value, but of different number and denomination. Blank paper, often sent from the United States to relatives in paper-short countries, was similarly replaced from a paper bank to obviate secret-ink transmissions. Childish scrawls, sent from proud parents to proud grandparents, were removed because of the possibility of their covering a map. Even lovers' X's, meant as kisses, were heartlessly deleted if censors thought they might be a code.

Censorship cable regulations prohibited sending any text that was unclear to the censor, including numbers unrelated to the text or a personal note in a business communication, and that was not in English, French, Spanish, or Portuguese plain language. To kill any possible sub rosa message, censors sometimes paraphrased messages. This practice gave rise to Censorship's classic tale, which dates back to World War I. Onto the desk of a censor

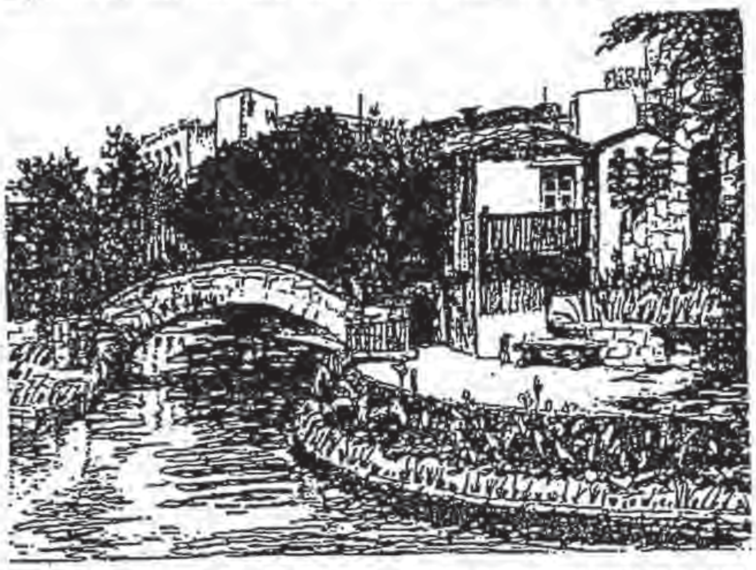
The second category of linguistically concealed messages is the semagram (from the Greek "sema," for "sign"). A semagram is a steganogram in which the ciphertext substitutes consist of anything but letters or numbers. The astragal of Aeneas the Tactician, in which yarn passing through holes representing letters carried the secret message, is the oldest known semagram. A box of Mah-Jongg tiles might carry a secret message. So might a drawing in which two kinds of objects represented the dots and dashes of Morse Code to spell out a message. The New York censorship station once shifted the hands and altered the positions of the individual timepieces in a shipment of watches lest a message be concealed in it.

The examination of the linguistically concealed messages—or, more correctly, those suspected to be such—was largely a frustrating experience. Often the examiner could not tell whether or not a message was hidden beneath the awkward or illiterate or misspelled writing. And even if he felt certain, solution often eluded him. He usually had only one message to work on, and no probable words. Early in the war, censorship practice even forbade working on a suspected cryptogram more than half an hour, on the theory that if the cryptanalyst hadn't gotten it by then, he'd never get it. These unsolved messages posed a difficult problem to the censors. Presumably they were carrying contraband information and so should be banned. But, in the absence of solution, no proof of this existed, and so the letter could not be mutilated. Sometimes this was done anyway, to destroy the suspected code.

Technological steganography early in the war consisted almost exclusively of invisible inks. This is truly an ancient device. Pliny the Elder, in his *Natural History*, written in the first century A.D., told how the "milk" of the tithymallus plant could be used as a secret ink. Ovid referred to secret ink in his *Art of Love*. A Greek military scientist, Philo of Byzantium, described the use of a kind of ink made from gall nuts (gallotannic acid), which could be made visible by a solution of what is now called copper sulfate. Qalqashandi described several kinds of invisible ink in his *Subh al-a' shā*. Alberti mentions them. The Renaissance employed them in diplomatic correspondence. About 1530 a book was printed with panels in invisible ink: if these pages were dipped in water, the message would appear; this could be repeated three or four times. Porta devoted Book XVI of his *Magia Naturalis* to invisible writing.

The common inks are of two kinds: organic fluids and sympathetic chemicals. The former, such as urine, milk, vinegar, and fruit juices, can be charred into visibility by gentle heating. Despite their antiquity and their minimal protection, they are so convenient that they were used even during World War II. Count Wilhelm Albrecht von Rautter, a naturalized American who was spying on his adoptive country for his native Germany, ran out of his good secret ink and had to use urine.

Sympathetic inks are solutions of chemicals that are colorless when dry but that react to form a visible compound when treated with another chemical, called the reagent. For example, when a spy writes in iron sulfate, nothing will be visible until it is painted over with a solution of potassium cyanate, when the two chemicals will combine to form ferric ferrocyanide, or Prussian blue, a particularly lovely hue. The colorless writing of lead sub-acetate will turn into a visible brown compound when moistened with sodium sulfhydrate. Copper sulfate can be developed with ammonia fumes, and it may have been this chemical that was used for the secret writing on the handkerchief of



A drawing of the San Antonio River that conceals a secret message (*solution in Notes*)

George Dasch, leader of the eight Nazi spies who landed by submarine on Long Island in 1942 to blow up American defense plants, railroad bridges, and canal locks. The red letters that appeared as if by magic when the pungent ammonia reached it spelled out the names and addresses of a mail drop in Lisbon and of two reliable sources for help in the United States. Each of the eight saboteurs had also been given a watertight tube containing four or five matchsticks tipped with a grayish substance that served as a ready-made pen-and-secret-ink. The trick in concocting a good secret ink is to find a substance that will react with the fewest possible chemicals—only one, if possible, thus resulting in what is called a highly "specific" ink.

To test for secret inks, censorship stations "striped" letters. The laboratory assistant drew several brushes, all wired together in a holder and each dipped in a different developer, diagonally across the suspected documents. The developers were wide-spectrum, picking up even such substances as body oils, so that fingerprints and sweat drops often showed up. On the other hand, they missed some specific inks. A bleaching bath removed the stripes. Letters were also checked by infrared and ultraviolet light. Writing in starch, invisible in daylight or under electric light, will fluoresce under ultraviolet. Infrared can differentiate colors indistinguishable in ordinary light and so can pick up, for example, green writing on a green postage stamp. The censorship field stations tested all suspicious letters and a percentage of ordinary mail picked at random, and sometimes all letters to and from a certain city for a week to see if anything suspicious turned up. During the war, about 4,600 suspicious letters were passed along to the F.B.I. and other investigative agencies; of these 400 proved to be of some importance.

Problems that would not yield to the crude approach of the field stations went back to the T.O.D. laboratory. Here, amid Bunsen burners and retorts, Pierce and Breon, aided by an expert photographer and laboratory technicians, cooked up reagents that would reincarnate the phantom writing. Better equipped and more deeply versed in the nuances of sympathetic inks than the mass-production workers of the field stations, they had received a great stimulus from contact with one of the great secret-ink experts of the world, England's Dr. Stanley W. Collins, who had conducted this battle of the test tubes in two World Wars: he spoke at the Miami Counter-Espionage Conference in August, 1943. T.O.D. soon learned that Nazi spies were taking countermeasures to frustrate the iodine-vapor test and the general reagent.

One was to split a piece of paper, write a secret-ink message on the inner surface, then rejoin the halves. With the ink on the inside, no reagent applied to the outside could develop it! The technique came to light when one German spy used too much ink and the excess soaked through. Sanborn Brown, the M.I.T. physicist, got two inmates of a local jail to explain how two sheets of parchment could be used to do the splitting. They had been caught misapplying the talent to one- and ten-dollar bills, pasting one half of the tens to one half of the ones and passing them with the ten-dollar side up. The method is more an art than a science, for if the sudden tear is not done just right, the paper will shred. To read the message, the paper must be resplit, but it comes apart much more easily the second time.

Another antidetection measure was transfer. German agents would write their message in invisible ink on one sheet of paper, then press this tightly against another sheet. Moisture in the air would carry some of the ink to the second sheet without the telltale differential wetting of the fiber papers on which the iodine test relied. This compelled T.O.D. to find the specific reagent required.

Perhaps the most interesting development of the secret-ink war was the German instrument discovered by Shaw, Pierce, and Richter in 1945 and dubbed the "Wurlitzer Organ" because of its resemblance to that musical instrument. They found a burned-out shell of one "organ" in the bombed remnants of the Munich censorship station, and an undamaged one in the censorship station on an upper floor of the Hamburg post office. It examined suspected letters on an assembly-line basis by ingeniously exploiting some principles of physics to make the invisible ink glow. It first exposed the paper to ultraviolet light. This pumped energy into chemicals of the ink, boosting their electrons out of their normal orbits into higher ones. The chemical was then in a metastable state. The heat from a source of infrared then nudged the electrons from their higher orbits back into their regular ones. As they did so, the substance would give up, in the form of visible light, the energy that it had absorbed from the ultraviolet. Since this phenomenon will occur for nearly all substances, even common salt, though some will naturally shine more brightly than others, the Germans had a system that would develop a good many inks.

The chief difficulty with secret inks was their inability to handle the great volume of information that spies had to transmit in a modern war. One way of channeling large amounts was to dot the meaningful letters in a newspaper with a solution of anthracene in alcohol. This was invisible under normal circumstances but glowed when exposed to ultraviolet light. But with newspapers being carried as third-class mail, this was hardly the fastest method of getting information to where it was going.

The Germans then came up with what F.B.I. Director J. Edgar Hoover called "the enemy's masterpiece of espionage." This was the microdot, a photograph the size of a printed period that reproduced with perfect clarity a standard-sized typewritten letter. Though microphotographs (of a lesser reduction) had carried messages to beleaguered Paris as far back as 1870, a tip to the F.B.I. in January of 1940 by a double agent, "Watch out for the dots—lots and lots of little dots," threw the bureau into a near panic. Agents feverishly looked everywhere for some evidence of them, but it was not until August of 1941 that a laboratory technician saw a sudden tiny gleam on the surface of an envelope carried by a suspected German agent—and carefully pried off the first of the microdots, which had been masquerading as a typewritten period.

At first the microdot process involved two steps: A first photograph of an espionage message resulted in an image the size of a postage stamp: the second, made through a reversed microscope, brought it down to less than 0.05 inches in diameter. This negative was developed. Then the spy pressed a hypodermic needle, whose point had been clipped off and its round edge sharpened, into the emulsion like a cookie cutter and lifted out the microdot. Finally the agent inserted it into a cover-text over a period and cemented it there with collodion. Later, one Professor Zapp simplified the process so that most of these operations could be performed mechanically in a cabinet the size of a dispatch case. The microdots, or "pats," as T.O.D. called them, were photographically fixed but were not developed; consequently, the image on them remained latent and the film itself clear. In this less obtrusive form they were pasted onto the gummed surface of envelopes, whose shininess camouflaged their own. The pats could show such fine detail because the aniline dye used as an emulsion would resolve images at the molecular level, whereas the silver compounds ordinarily used in photography resolve only down to the granular level.

The microdots solved the problem of quantity flow of information for the Nazis. Professor Zapp's cabinets were shipped to agents in South America, and soon a flood of material was being sent to Germany disguised as hundreds of periods in telegraph blanks, love letters, business communications, family missives, or sometimes as a strip of the tiny film hidden under a stamp. The very first discovered, and the most frightening, was one in which a spy was asked to discover "Where are being made tests with uranium?" at a time when the United States was fighting to keep secret its development of the atom bomb. The "Mexican microdot ring," which operated from a suburb of Mexico City, microphotographed trade and technical publications that were barred from international channels—a favorite was *Iron Age*, with statistics on American steel production—and sent them to cover addresses in Europe on a wholesale basis, with as many as twenty pats in a single letter. Technical drawings also went by microdot. Other microdots talked of blowing up seized Axis ships in southern harbors, the deficient condition of one of the Panama Canal locks, and so on. Censorship discovered many of these, now that it knew what to look for, and this enabled the F.B.I.'s wartime Latin American branch to break up one Axis spy ring after another.

With mail and cable routes being screened so closely and subject to unpredictable delays, it was not unlikely that Axis agents would take to the ether to gain speed and avoid censorship. But here, too, the United States was ready for them.

The Radio Intelligence Division of the Federal Communication Commission had the job, in peacetime, of policing the airwaves, which are public property, for violations of federal radio regulations. During the war, its 12 primary and 60 subordinate monitoring posts and about 90 mobile units patrolled the radio spectrum for enemy agent radios. Teletype linked them into a direction-finding net coordinated from Washington. R.I.D. employed the latest radio equipment, including an aperiodic receiver that would give an alarm whenever it picked up a signal on any of a wide range of frequencies, and the "snifter," a meter that a man could carry in the palm of his hand while inspecting a building to see which apartment a signal came from.

> In the routine day-and-night operation of a monitoring station [wrote George E. Sterling, R.I.D.'s chief], the patrolman of the ether would cruise his beat, passing up and down the frequencies of the usable radio spectrum, noting the

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

MAR-14-97 15:55 From:ELECTRICAL ENGINEERING 6126254583 T-717 P.01/05 Job-759

Post-it® Fax Note 7671	Date	# of pages ▶ 5
To E. Koblenz	From	
Co./Dept.	Co.	
Phone #	Phone #	
Fax # 202-429-0736	Fax #	

5

;NAL PROCESSING VI
THEORIES AND APPLICATIOI

Proceedings of EUSIPCO-
Eighth European Signal Processing Conferer
Trieste, Italy, 10-13 September 1!

Editec
G. RAMP
G.L. SICURAI
S. CARR
S. MA

D.E.I
University of Trieste, I

VOLUME

EU

Edizioni LINT T

BEST AVAILABLE COPY

Digital Watermarks for Audio Signals *

Laurence Boney
Departement Signal, ENST, Paris, France 75634
email: boney@email.enst.fr

Ahmed H. Tewfik and Khaled N. Hamdy
Department of Electrical Engineering, University of Minnesota, Minneapolis, MN 55455
email: tewfik@ee.umn.edu, khamdy@ee.umn.edu

ABSTRACT

In this paper, we present a novel technique for embedding digital "watermarks" into digital audio signals. Watermarking is a technique used to label digital media by hiding copyright or other information into the underlying data. The watermark must be imperceptible and should be robust to attacks and other types of distortion. In addition, the watermark also should be undetectable by all users except the author of the piece. In our method, the watermark is generated by filtering a PN-sequence with a filter that approximates the frequency masking characteristics of the human auditory system (HAS). It is then weighted in the time domain to account for temporal masking. We discuss the detection of the watermark and assess the robustness of our watermarking approach to attacks and various signal manipulations.

1 Introduction

In today's digital world, there is a great wealth of information which can be accessed in various forms: text, images, audio, and video. It is easy to ensure the security of "analog documents" and protect the author from having his work stolen or copied. The question is *how do you copyright or label digital information and preserve its security without destroying or modifying the content of the information.*

Data hiding, or steganography, refers to techniques for embedding watermarks, signatures, and captions in digital data. A watermark could be used to provide proof of "authorship" of a signal. Similarly, a signature is used to provide proof of ownership and track illegal copies of the signal. Watermarking is an application which embeds a small amount of data, but requires the greatest robustness because the watermark is required for copyright protection [1]. One approach to data security is to use encryption [1]; however, once the documents are decrypted, the "signature" is removed and there is *no proof of ownership* such as a label, stamp, or watermark. Note that data hiding does not restrict access to the original information as does cryptography.

The watermark should: be inaudible [1, 2]; be statistically invisible to prevent unauthorized detection and/or removal by "pirates"; have similar compression characteristics as the original signal to survive compression/decompression operations; be robust to deliberate attacks by "pirates"; be robust to standard signal manipulation and processing operations on the host data, e.g., filtering, resampling, compression, noise, cropping, A/D-D/A conversions, etc; be embedded directly in the data, not in a header; support multiple watermarkings; be self-clocking for ease of detection in the presence of cropping and time-scale change operations.

Observe that a "pirate" can defeat a watermarking scheme in two ways. He may manipulate the audio signal to make the watermark undetectable. Alternatively, he may establish that the watermarking scheme is unreliable, e.g., that it produces too many false alarms by detecting a watermark where none is present. Both goals can be achieved by adding inaudible jamming signals to the audio piece. Therefore, the effectiveness of a watermarking scheme must be measured by its ability to detect a watermark when one is present (probability of detection) and the probability that it detects a watermark when none is present (probability of a false alarm) in the presence of jamming signals and signal manipulations.

Several techniques for data hiding in images have been developed [1, 3, 4, 5, 6]. A method similar to ours is proposed in [2], where the *N* largest frequency components of an image are modified by Gaussian noise. However, the scheme only modifies a subset of the frequency components and does not take into account the human visual system (HVS). The audio watermark we propose here embeds the *maximum* amount of information throughout the spectrum while still remaining perceptually inaudible. It is well-known that detection performance improves with the energy of the signal to be detected. Therefore, we effectively improve the performance of the watermarking scheme by increasing the energy of the watermarked signal while keeping it inaudible.

In [7, 8], we presented a novel technique for embedding digital watermarks into audio signals. Note that our approach is similar to that of the approach of [1], in that we shape the frequency characteristics of a PN-sequence. However, unlike [1] we use perceptual masking models of the HAS to generate the watermark. In particular, our scheme for audio is the only one that uses the frequency masking models of the HAS along with the temporal masking models to hide the copyright information in the signal. We also provide a study of the detection performance of our watermarking scheme. Our results indicate that our scheme is robust to lossy coding/decoding, D/A - A/D conversion, signal resampling, and filtering. In this paper, we present further results showing that our scheme is robust when the watermarks which are

*This work was partially supported by AFOSR under grant AF/F49620-94-1-0461 and by NSF under grant NSF under grant NSF/INT-9406954.

EUSIPCO-96 pp 1697-17

composed of multiple PN-sequences and is robust in the presence of audible distortions due to vector quantization.

Finally, observe that the approach described here for watermarking audio signals can also be used to watermark image and video data with appropriate modifications and extensions (c.f. [7, 9]).

2 Watermark Design

Each audio signal is watermarked with a unique codeword. Our watermarking scheme is based on a repeated application of a basic watermarking operation on processed versions of the audio signal. The basic method uses three steps to watermark an audio segment as shown in Fig. 1. The complete watermarking scheme is shown in Fig. 2. Below we provide a detailed explanation of the basic watermarking step and the complete watermarking technique.

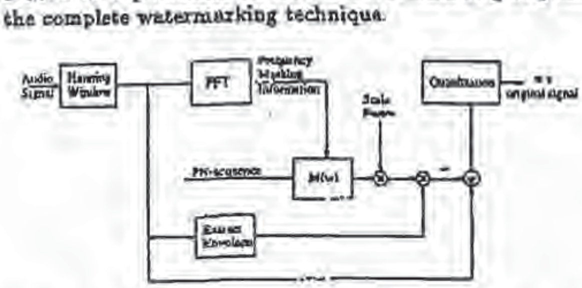


Figure 1: Watermark Generator: First stage for audio

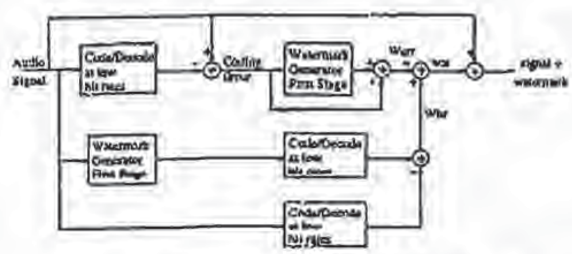


Figure 2: Full Watermark Generator for audio

2.1 The basic watermarking step

The basic watermarking step starts with a PN-sequence. Maximum length PN-sequences are used in our watermarking scheme because they provide an easy way to generate a unique code for an author's identification. Like random binary sequences, PN sequences have 0's and 1's that occur with equal probabilities. The autocorrelation function (ACF) of such a sequence has period N and is binary valued [10]. Because of the periodicity of the ACF, the PN sequence is self-clocking. This allows the author to synchronize with the embedded watermark during the detection process. This is important if the signal is cropped and resampled.

To generate the watermark, we first calculate the masking threshold of the signal using the MPEG Audio Psychoacoustic Model 1, [11]. The masking threshold is determined on consecutive audio segments of 512 samples. Each segment is weighted with a Hanning window. Consecutive blocks overlap by 50%. The masking threshold is then approximated with a 10^{th} order all-pole filter, $M(\omega)$, using a least squares criterion. The PN-sequence, $seq(\omega)$, is filtered with the approximate masking filter, $M(\omega)$, in order to ensure that the spectrum of the watermark is below the masking threshold.

Since the spectral content of the audio signal changes with time, watermarks added to different blocks will be in general different even if they are generated from the same starting PN-sequence. However, it is preferable to use different PN-sequences for different blocks to make the statistical detection by an unauthorized user of the watermark more difficult. Note also that using long PN-sequences or embedding long cryptographic digital signatures also helps in that respect.

Frequency domain shaping is not enough to guarantee that the watermark will be inaudible. Frequency domain masking computations are based on Fourier analysis. A fixed length FFT does not provide good time localization for our application. In particular, a watermark computed using frequency domain masking will spread in time over the entire analysis block. If the signal energy is concentrated in a time interval that is shorter than the analysis block length, the watermark is not masked outside of that subinterval. This then leads to audible distortion, e.g., pre-echoes. To address this problem, we weight the watermark in the time domain with the relative energy of the signal.

The time domain weighting operation attenuates the energy of the computed watermark. In particular, watermarks obtained as above have amplitudes that are typically smaller than the quantization step size. Therefore, the watermark would be lost during the quantization process. Note also that, as observed earlier, detection performance is directly proportional to the energy of the watermark. We have found that it is possible to prevent watermark loss during quantization and improve detection performance by amplifying the watermark by 40 dB before weighting it in the time domain with the relative energy of the signal. We have found experimentally that this amplification does not affect the audibility of the watermark because of the attenuation effect of the time domain weighting operation.

2.2 The full watermarking scheme

As mentioned above, the watermarking scheme must be robust to coding operations. Low bit rate audio coding algorithms tend to retain only the low frequency information in the signal. We, therefore, need to guarantee that most of the energy of the watermark lies in low frequencies. After experimenting with many schemes, we have found that the best way to detect the low frequency watermarking information is to generate a low-frequency watermark as the difference between a low bit rate coded/decoded watermarked signal and the coded/decoded original signal at the same bit rate. Watermarking is done using the basic watermarking step described above. The low bit rate chosen to implement this operation is the minimal bit rate for which near-transparent audio coding is known to be possible for signals sampled at the rate of the original signal. This scheme is more effective than other schemes that attempt to add the watermark on a lowpass filtered version of the signal because the coding/decoding operation is not a linear and does not permute with the watermarking operation. Fig. 2 illustrates the above procedure for signals sampled at an arbitrary sampling rate. The low-frequency watermarking signals is shown as w_{br} in Fig. 2. Here, the subscript br refers to the bit rate of the coder/decoder.

For best watermark detection performance at higher bit rates, we need to add watermarking information in the higher frequency bands. We do so by producing a watermark w_{err} for the coding error. The coding error is the difference between the original audio signal and its low bit rate coded version. The watermark w_{err} is computed using the basic watermarking step described at the beginning of this section. The final watermark is the sum of the low-frequency watermark and the coding error watermark.

2.3 Listening tests: audibility of the watermarks

We used segments of four different musical pieces as test signals throughout the experiment: the beginning of the third movement of the sonata in B flat major D 960 of Schubert, interpreted by Vladimir Ashkenazy, a castanet piece, a clarinet piece, and a segment of "Tom's Diner" an *a capella* song by Suzanne Vega (svega). The Schubert signal is sampled at 32 kHz. All other signals are sampled at 44.1 kHz. Note that the castanets signal is one of the signals prone to pre-echoes. The signal svega is significant because it contains noticeable periods of silence. The watermark should not be audible during these silent periods.

The quality of the watermarked signals was evaluated through informal listening tests. In the test, the listener was presented with the original signal and the watermarked signal and reported as to whether any differences could be detected between the two signals. Eight people of varying backgrounds, including the authors, were involved in the listening tests. One of the listeners had the ability to perceive absolute pitch and two of the listeners had some background in music.

In all four test signals, the watermark introduced no audible distortion. No pre-echoes were detected in the watermarked castanet signal. The quiet portions of svega were similarly unaffected.

3 Detection of the Watermark

Let us now describe the watermark detection scheme and the detection results that we have obtained. In the experimental work described below, we used shaped inaudible noise to simulate attacks by pirates and distortions due to coding. We also tested the effects of filtering, coding, D/A - A/D converting and re-sampling on the detection performance of the proposed scheme. The detection results that we report below are based on processing 100 blocks of the observed signal of 512 samples. Note that this corresponds to 1.6 sec at the 32 kHz sampling rate and 1.16 sec at the 44.1 kHz sampling rate.

Our detection scheme assumes that the author has access to the original signal and the PN-sequence that he used to watermark the signal. It also assumes that the author has computed the approximate bit rate of the observed audio sequence $r(k)$. To decide whether the given signal $r(k)$ has been watermarked or not, the author subtracts from $r(k)$ a coded version s_{br} of the original audio signal $s(k)$. The signal s_{br} is produced by coding $s(k)$ at the estimated bit rate of $r(k)$ using the MPEG coding procedure. Note that $r(k)$ itself may have been coded using a different coding algorithm. The difference between the output of the MPEG coding algorithm operating on the original signal at the estimated bit rate and that of the actual coding algorithm at the true bit rate will appear as an additive noise signal.

Next, the author needs to solve the following hypothesis testing problem:

- $H_0 : x(k) = r(k) - s_{br}(k) = n(k)$
- $H_1 : x(k) = r(k) - s_{br}(k) = w'(k) + n(k)$.

Here, $n(k)$ denotes an additive noise process that includes errors due to different coding algorithms and signal manipulations, intentional jamming signals and transmission noise. The signal, $w'(k)$, is the modified watermark. Since the precise nature of $n(k)$ is unknown, we solve the above hypothesis testing problem by correlating $x(k)$ with $w'(k)$ and comparing the result with a threshold. Note that one needs to estimate time-scale modifications prior to correlations if such modifications have been performed on the signal. Fig. 3 shows the result of correlating a watermark corresponding to a segment of the Schubert audio piece with itself, the jammed watermark corrupted by frequency shaped noise of maximum masked intensity and shaped noise of maximum masked intensity alone. In all cases, the signal was not coded. The figure clearly indicates that reliable detection is feasible.

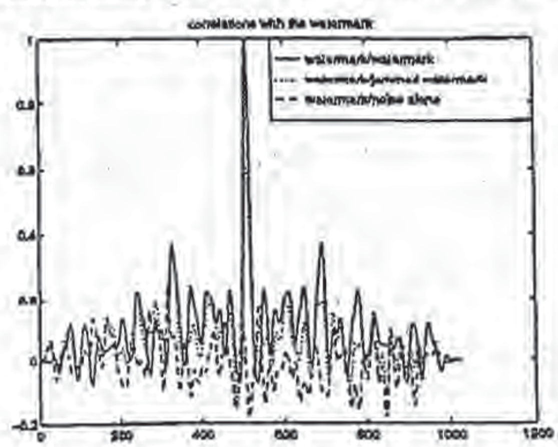


Figure 3: Detection of the watermark in Schubert with additive noise

3.1 Generation of the Additive Noise

Noise which has the same spectral characteristics as the masking threshold provides an approximation of the worst possible additive distortion to the watermark. This type of distortion is a good worst case model for distortions due to intentional jamming with inaudible signals and mismatches between the actual and assumed coding algorithms.

The noise that we have used in our experiments was generated in the same way as the watermark. Specifically, the masking threshold is first shifted $+40dB$ and multiplied by the discrete Fourier transform of a Gaussian white noise process. The resulting noise is then weighted in time by the relative energy of the signal. After quantization, we filter this shaped noise by the masking threshold and requantize it. The resulting noise is almost completely inaudible and is a good approximation of the maximum noise that we can add below the masking threshold.

3.2 Summary of Detection Results

Let us now summarize the detection results that we have obtained. Each group of results is meant to illustrate the robustness of our approach to a specific type of signal manipulation.

Robustness to MPEG coding for single and multiple watermarks
To test the robustness of our watermarking approach to coding, we added noise to several watermarked and non-watermarked audio pieces and coded the result. The watermarks were generated by using different PN sequences in different audio segments. The noise was almost inaudible and was generated using the technique described above. The coding/decoding was performed using a software implementation of the ISO/MPEG-1 Audio Layer III coder with several different bit rates. We then attempted to detect the presence of the watermark in the decoded signals. Table 1 shows that P_{detect} is 1 or nearly 1 in all cases and $P_{falsealarm}$ is nearly 0 in all cases.

We reported in [8] other detection results corresponding to an earlier implementation of our watermarking scheme that used the same PN sequence to watermark all segments of an audio piece. We also reported in that reference the results of detecting multiple watermarks added to a single audio piece. There are many instances where it is useful to add multiple watermarks to a signal. For example, there may be multiple authors for a piece of music, each with his/her own unique id. When detecting specific watermark, the other watermarks are considered to be noise. The results of [8] indicate that with one or more watermark, P_{detect}, is 1 or nearly 1 in all cases. Equally important, the probability of false alarm, $P_{falsealarm}$ is nearly 0 in all cases. These results, together with the ones presented here, establish the robustness of our scheme to MPEG coding and multiple watermarking.

Robustness to VQ distortion
We also tested the robustness of our watermarking approach to VQ coding. The codebooks consisted of 16 bit codewords. The audio signals were processed through codebooks of various sizes: 64, 128, 256, and 512 codewords. Although the signal was noticeably distorted, the watermark detection was unaffected, as shown in Table 2: P_{detect} is 1 or nearly 1 in all cases and $P_{falsealarm}$ is nearly 0 in all cases.

In [8], we also show that our watermarking scheme is robust to signal resampling. We are currently assessing the robustness of our scheme to time-scale modifications of the signal.

4 Conclusions

Our method for the digital watermarking of audio signals extends the previous work on images. Our watermarking scheme consists of a maximal length PN-sequence filtered by the approximate masking characteristics of the HAS and weighted in time, our watermark is imperceptibly embedded into the audio signal and easy to detect by the author thanks to the correlation properties of PN-sequences. Our results show that our watermarking scheme is robust in the presence of additive noise, lossy coding/decoding, VQ distortion, multiple watermarks, resampling, and time-scaling.

References

[1] W. Bender, D. Gruhl, and N. Morimoto, "Techniques for data hiding," *Proc. of the SPIE*, 1995.

[2] I. Cox, J. Kilian, T. Leighton, and T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia." Tech. Rep. 95-10, NEC Research Institute, 1995.

[3] O. Bruyndonckx, J.-J. Quisquater, and B. Macq, "Spatial method for copyright labeling of digital images," in *Nonlinear Signal Processing Workshop, Thessaloniki, Greece*, pp. 456–459, 1995.

[4] I. Pitas and T. Kaskalis, "Applying signatures on digital images," in *Nonlinear Signal Processing Workshop, Thessaloniki, Greece*, pp. 460–463, 1995.

[5] E. Koch and J. Zhao, "Towards robust and hidden image copyright labeling," in *Nonlinear Signal Processing Workshop, Thessaloniki, Greece*, pp. 452–455, 1995.

[6] P. Boland, J. O'Ruanaidh, and C. Dautzenberg, "Watermarking digital images for copyright protection," *IEE Intl. Conf. on Image Proc. and Its Apps., Edinburgh, 1995.*

[7] L. Boney, A. Tewfik, K. Hamdy, and M. Swanson, "Digital watermarks for multimedia." Submitted to U.S. Patent Office, February 1996.

[8] L. Boney, A. Tewfik, and K. Hamdy, "Digital watermarks for audio signals," in *IEEE Int. Conf. on Multimedia Computing and Systems*, (Hiroshima, Japan), June 1996.

[9] M. Swanson, B. Zhu, and A. Tewfik, "Transparent robust image watermarking," in *to appear ICIP'96*, (Lausanne, Switzerland), Sept. 1996.

[10] S. Haykin, *Communication Systems, 3rd Edition*. John Wiley and Sons, 1994.

[11] "Codage de l'image animee et du son associe pour les supports de stockage numerique jusqu'a environ 1,5 mbit/s," tech. rep., ISO/CEI 11172, 1993.

Table 1: Multiple PN sequence watermark with MPEG distortion

Bit Rate	Watermark	*Schubert*	*Clarinet*	*Castanet*
kbits/sec	Threshold	0.65	0.47	0.54
48	P_{detect}	0.9922	na	na
	$P_{falsealarm}$	0.0117	na	na
64	P_{detect}	0.9961	1	1
	$P_{falsealarm}$	0	0	0.0031
128	P_{detect}	1	1	1
	$P_{falsealarm}$	0	0	0
160	P_{detect}	1	1	1
	$P_{falsealarm}$	0	0	0
224	P_{detect}	1	1	1
	$P_{falsealarm}$	0	0	0
320	P_{detect}	na	1	1
	$P_{falsealarm}$	na	0	0
	# of trials	257	83	639

Table 2: Watermark detection with VQ distortion

Bit Rate	Signal	*Clarinet*	*Castanet*	*Svego*
bits/sample	Threshold	0.54	0.46	0.52
6	P_{detect}	1	1	1
	$P_{falsealarm}$	0	0.0087	0
7	P_{detect}	1	1	1
	$P_{falsealarm}$	0.0010	0.01	0
8	P_{detect}	1	1	1
	$P_{falsealarm}$	0	0.0007	0
9	P_{detect}	1	0.9997	1
	$P_{falsealarm}$	0	0.0890	0
	# of trials	3000	3000	3000

(11)

Copy Protection for Multimedia Data based on Labeling Techniques

G.C. Langelaar, J.C.A. van der Lubbe, J. Biemond

Department of Electrical Engineering, Information Theory Group
Delft University of Technology
P.O.Box 5031, 2600 GA Delft, The Netherlands
E-mail: *{Gerhard, vdLubbe}@it.et.tudelft.nl*

Abstract
Service providers are reluctant to distribute their multimedia data in digital form because of their fears for unrestricted duplication and dissemination. Therefore robust methods must be developed to protect the proprietary rights of the multimedia data owners and to realize a copy protection mechanism.
In this paper the existing methods for labeling multimedia data are discussed and evaluated. A possibility is given to extend these methods towards a robust copy protection system for new mass storage devices. Some methods suitable for a copy protection method are selected and weak points are discussed. One of these methods is extended to embed a bit sequence instead of one bit in an image and to make it more resistant to lossy compression techniques. Using this method some true color images (size about 500 x 500) were labeled with about 200 bits. The label turned out to be resistant to JPEG compression, with quality parameter set up to 40% (compression rate >1:20).

1. Introduction

Nowadays digital recording devices are available for recording audio. Using personal computers it is also possible to store digital video on a harddisk. The storage capacity of a harddisk is, however, not sufficient to store a complete full resolution home video. For the consumer it would be easier to have one digital storage device that can handle huge amounts of multimedia data. Such a device can replace all other recording equipment in the home, like tape or DAT recorder, VCR and tape streamer.
The aim of the SMASH project , supported by several companies and universities, is to develop a popular mass-home-storage-device. The development rate of such a digital mass storage system is dependent on not only technical advances, but also on the existence and evolution of adequate protection methods on it. Therefore, robust methods must be developed to protect the proprietary rights of the data owners and to realize a copy protection mechanism limiting the easiness of duplication of multimedia data.
A copy protection system called SCMS [1] (Serial Copy Management System) exists for digital audio recorders, like the DAT, DCC and minidisk recorders. Using this system, a consumer can make only one digital copy of any digital source. Such a copy can not be duplicated further using storage devices equipped with this protection method.
The protection is embedded in the transfer protocol. Together with the music data some sub-code data is transmitted. One bit in this sub-code is called the copy prohibit bit. This bit is set to "one" for every recording. If the consumer tries to record audio data containing a copy prohibit bit, the storage device

is subtracted from the original one. If the mean of a block of pixel differences exceeds a certain threshold, the corresponding bit is taken as '1', otherwise as '0'. After JPEG compression, with quality parameter set to 30%, the label can still be recovered. A disadvantage of this method is that the original unlabeled image is required to decode the label.
Zhao and Koch [8] propose a method to embed a bitstream in the DCT domain. The image is divided up into 8x8 blocks (like the JPEG algorithm does). From pseudo-random selected 8x8 blocks the DCT coefficients are calculated. These coefficients are quantized using a quality factor Q and the standard quantization matrix of the JPEG software. Three quantized coefficients are selected and adapted in such a way that they have a certain order in size. For example if a bit '1' must be embedded in a block, the third coefficient must be smaller than the other two. In an earlier proposal by the same authors [9], two instead of three coefficients were used. After JPEG compression, with quality parameter set to 50%, the label can still be recovered. Advantages of this method are that the original unlabeled image is not required to decode the label and that a quite large bitstream can be embedded.
Cox *et al.* [10] embed a sequence of real numbers of length n in an N x N image by computing the N x N DCT and adding the sequence to the n highest DCT coefficients, excluding the DC component. To extract the sequence, the DCT transform of the original image is subtracted from the DCT transform of the labeled one and the sequence is extracted from the highest coefficients. A disadvantage is that the original unlabeled image is required to decode the label.
Boland *et al.* [11] describe a method that works with different image transforms (DCT, Walsh-Hadamard, Wavelet, Fast Fourier). An image is divided into blocks, the mean of the block is subtracted from each pixel in the block and the remaining values are normalized between -127 and 127. The transform is carried out on the image block and some coefficients are modulated to embed a number of bits, for instance by adding one to a coefficient for bit '1' or subtracting one for bit '0'. A reverse transformation is carried out and the original block is replaced by the labeled one. A disadvantage of this method is that the original unlabeled image is required to decode the label. After JPEG compression, with quality parameter set to 90%, a label could be recovered from an image with a bit error rate of 14% using the DCT transform technique, using other transforms the bit error rates were higher.

3. Evaluation labeling methods

The labeling methods described above can add information to an image in an invisible way, but there is always a trade-off between the size of the label, the resistance to JPEG compression and the effect on the image quality, although estimating the quality degradation due to labeling is a completely subjective matter.
The methods, that add the label in the spatial domain, seem to have the lowest bit capacity and the lowest resistance to JPEG compression (methods of Bender, Pitas and Caronni).
Adding the label in another domain sometimes improves the capacity and the resistance. The use of the DCT transform gives the best results (methods of Zhao, Cox and Boland), obviously because the JPEG algorithm makes use of the same DCT transform. The resistance can be increased further if the quantization step is also taken into account (method of Zhao).
If the original unlabeled image can be used together with the labeled one to extract the label, the capacity and the resistance to JPEG compression seem to be higher (methods of Caronni and Cox). In the latter case, the method is also more robust to other attacks, like cropping, rotation, translation, scaling etc. Using the original image some preprocessing can be done before the label is checked. Rotation angles, translation and scale vectors can be estimated and missing parts of the image can be replaced by parts from the original image.

simply refuses to record.
However, this system can not be applied on a mass storage system for multimedia data with several interfaces to different kinds of devices. For example, if data is transferred from the storage device to the harddisk of a personal computer, the copy prohibit bit is lost, because it was only part of the transfer protocol. Therefore, the copy prohibit bit needs to be directly encoded into the audio, image or video signal itself. In this way the bit remains intact across varying data file formats. It is obvious that the copy prohibit bit must be inaudible or invisible for the user and that it must be difficult to remove the bit by using lossy compression techniques, filtering or other processing techniques, that change the data, but do not considerably affect the quality of the data.
By embedding a copy prohibit bit in the data it is still possible to copy the data to devices that are not equipped with this copy protection system. However, if data is copied to such devices it can not directly be played back due to a lower transfer rate of that recording device (e.g. tape streamer) or the amount of data is too big to fit because of a limited storage capacity. So, a few images out of a digital library or some audio fragments can be copied, but it is probably more expensive and time consuming to store this data on other media than buying the original data and using the new mass storage device.
In this paper the existing methods for labeling multimedia data are discussed and evaluated. After that, some methods suitable for a copy protection method as described above are selected and weak points are discussed. One of these methods is extended to embed a bit sequence instead of one bit in an image and to make it more resistant to lossy compression techniques. Finally, conclusions are drawn.

2. Existing Copyright Labeling Techniques

The technique of embedding information in image, video and data is called steganography. It is mainly used in the field of copyright labeling, where data is labeled to identify it uniquely as property of the copyright holder. A label normally consists of a binary serial number or an ASCII text string. Several projects are working or have worked on this subject, like the EC RACE project ACCOPI [2] and the ACTS project TALISMAN [3].
Labels can be added in almost every domain (Spatial, DCT, Wavelet, Fourier, etc.) using different methods. There are two possibilities to extract the label from the image, some methods only use the labeled image, others also uses the original image. The simplest method manipulates the least significant bit of the luminance values or color components of an image, in a manner which is undetectable to the eye [4]. However, this method is not resistant to for instance JPEG compression.
The two following methods embed a label of one bit in the spatial domain. Bender *et al* [5] describe a statistical labeling method called "Patchwork". Using this method, n pairs of image points (ai, bi) are randomly chosen. The brightness of ai is increased by one and the brightness of the corresponding bi is decreased by one. The expected value of the sum of the differences of the n pairs of points is then $2n$. The authors show that after JPEG compression, with quality parameter set to 75%, the label can still be decoded with a probability of recovery of 85%.
Pitas and Kaskalis [6] describe a similar method. Using this method the picture is split in two subsets of equal size (for example by using a random generator) and the brightness of the pixels of one subset is altered by adding a positive integer factor k. This factor k is calculated using the sample variances of the two subsets. To check the label the difference between the means of the two subsets of pixels is calculated. The expected value is k if a label was added. This method is only resistant to JPEG compression ratios up to 4:1 (quality factor of more than 90%). The major drawback of these two methods is the extremely low bit capacity, usually one bit.
Caronni [7] also describes a method which embeds a bitstream in the luminance values of an image. The image is divided up into blocks. Every pixel in a block is incremented by a certain factor to encode a '1' and is left untouched to encode a '0'. To recover a label, the brightness of each pixel in the labeled image

4. Suitable methods for a copy protection system

For the copy protection system described in the introduction, the following requirements must be met:

- The method must have a bit capacity of at least 1 bit, but a bit capacity up to a few hundred bits is preferable, because of extra options like adding timestamps.
- It must be possible to extract the embedded code without using the original unlabeled data.
- The label must be resistant to lossy compression techniques (like JPEG / MPEG), filtering or other processing techniques, that change the data, but do not considerably affect the quality.
- The labeling is allowed to cause degradation of the quality of the data if the data was already labeled before. Normally data is labeled only once. But if a hacker changed for example the image by a slight translation or rotation, the storage device might be unable to read out the original label and deals with the data as new unlabeled data. The new label should now affect the quality.

The only methods which meet these requirements, are the methods of Bender, Pitas and Zhao. However, from these three methods only the last one (Zhao) has a sufficient bit capacity and an acceptable resistance to JPEG compression, the other two must be developed further to achieve the same results. In the next section a proposal is given to extend one of the first methods.
A weak point of the method of Zhao is that the quality of the picture is heavily reduced by a label, which is resistant to JPEG compression up to a quality of 50%. This is illustrated in Figure 1. In the left half of the picture (1a) the unlabeled image and a corresponding zoom view of the shoulder is given. In the right half (1b) the labeled image (quality 50%) and the corresponding zoom view of the shoulder is represented.

Figure 1a. Unlabeled image and zoom view **Figure 1b.** Labeled image using Zhao's method

If bits are added with a certain quality factor, the quality of many parts in the image (a number of 8x8 blocks) is reduced. Another disadvantage of this method and also of the methods of Bender and Pitas is that the labeling techniques are not resistant to attacks like cropping, rotation, translation and scaling.

5. Extending spatial labeling method

In this section a new block based method is proposed, which adds a bit sequence in the spatial domain. This method is based on the method of Pitas described in the previous section.
Different variants of this method have been tested, but the method as described below gave the best experimental results (concerning the resistance to JPEG compression).

Labeling procedure:
A label consists of a few hundred bits. Each label bit is embedded in a block of luminance values. The width and height of this block are multiples of 8. The X and Y positions of the top corner of the block in the image are also multiples of 8 to be compatible with the YUV based JPEG compression algorithm (e.g. the JFIF standard).

1. First the RGB color image is converted to the YUV domain.

2. A block **B** is pseudo-randomly selected from the image to embed one label bit.

3. A fixed binary pseudo-random pattern of the same size as the block is generated, consisting of the integers **"0"** and **"1"**.

4. The mean **Io** is calculated of the luminance values in the block, where the random sequence is 0. The mean **I1** is calculated of the luminance values in the block, where the random sequence is 1. After that, the difference **Difference_High_Quality_Block(I0,I1)** is calculated between the two means.

5. In a similar way, the difference **Difference_Low_Quality_Block(I'0,I'1)** is calculated for a copy **B'** with reduced quality of the block **B** by taking the 8x8 DCT transform, quantizing the coefficients with a certain quality factor **Q** followed by an inverse DCT transform.

6. If **label bit "1"** must be embedded skip step 7.

7. In order to embed the **label bit "0"**, the integer random pattern is subtracted from the original block, if one of the two differences exceeds the value zero. The procedures (4,5,7) are repeated iteratively until both differences are below zero. Step 8 is skipped.

8. In order to embed **label bit "1"**, the integer random pattern is added to the original block, if one of the two differences is smaller than a certain threshold **T**. The procedures (4,5,8) are repeated iteratively until both differences exceed **T**.

9. The procedures (2..8) are applied to all pseudo-randomly selected blocks until all bits of the label are embedded.

10. Finally the YUV values are converted to the RGB domain.

The algorithm is more robust to JPEG (JFIF) compression, if a higher threshold **T** and a lower quality factor **Q** is chosen.

Label extracting procedure:
Reading out the label is simple and is described below.

1. First the RGB color image is converted to the YUV domain.

2. A block **B** is pseudo-randomly selected from the image to read out one bit.

3. The fixed binary pseudo-random pattern of the same size as the block is generated, consisting of the integers **"0"** and **"1"**.

4. The mean **Io** is calculated of the luminance values in the block, where the random sequence is 0. The mean **I1** is calculated of the luminance values in the block, where the random sequence is 1. After that, the difference **Difference(I0,I1)** is calculated between the two means.

5. If this difference **Difference** exceeds the value zero the bit embedded in the block is one, otherwise zero.

6. The procedures (2..5) are applied to all pseudo-randomly selected blocks until all bits of the label are extracted.

Applying a simple edge-enhance filter to the luminance pixel values before-checking the label reduced the percentage of bit errors considerably. The method can further be improved by adapting the random pattern. If the ratio between the numbers of ones and zeros in the random pattern is forced to be 1:4 the labeling is significantly less visible to the human eye, but marginally weaker. If the dotsize of the random pattern is increased to 2x2 instead of 1x1, the robustness increases.

6. Experimental results

Using the method described in the previous section four color images were labeled (see table 1 for more information about these images). The ratio between the numbers of ones and zeros in the random pattern was forced to be 1:4 and the pattern dotsize was adapted as described in the previous section. Each bit was embedded in a block of 32 x 32 pixels, the threshold T was set to 1 and a quality factor of 75% was used.

Table 1. Information about the *labeled* test images.

Name	Resolution (pixels)	Compression ratio using JPEG quality factor of 90%	80%	75%	60%	50%	40%
Diver	302 x 323	1:9	1:14	1:16	1:23	1:27	1:30
Mountain	733 x 487	1:8	1:11	1:12	1:18	1:21	1:25
Lena	512 x 512	1:11	1:17	1:20	1:28	1:33	1:39
Kielp	720 x 576	1:7	1:10	1:12	1:16	1:18	1:21

In Table 2, Figure 2 and 3 the bit errors in the label are represented, after compressing the images with the JPEG compression algorithm, with quality parameter set to different values.

Table 2. Number of bit errors after JPEG compression *(without / with edge enhance filtering)*.

Name	Label length	bit errors after JPEG compression with quality factor of 90%	80%	75%	60%	50%	40%

Diver	90 bits	0 / 0	2 / 0	2 / 0	7 / 2	17 / 9	15 / 7
Mountain	208 bits	20 / 5	11 / 3	3 / 0	23 / 9	19 / 5	43 / 23
Lena	208 bits	1 / 0	5 / 0	6 / 0	30 / 1	37 / 5	50 / 10
Kielp	208 bits	0 / 0	1 / 1	5 / 2	16 / 6	25 / 13	33 / 15

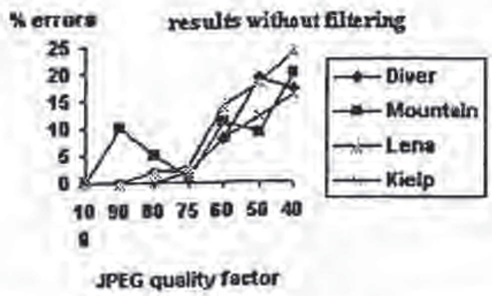


Figure 2.% bit errors after JPEG compression withhout using edge-enhance-filtering

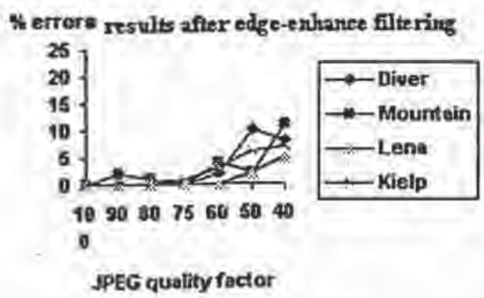


Figure 3.% bit errors after JPEG compression using edge-enhance-filtering

Applying a simple edge enhance filter improves the results considerably, because it amplifies the differences of the adapted and the unaffected luminance values. The maximal percentage of bit errors in the label is only 11% after compressing the image using a quality factor of 40%. Heavy smoothing, obviously, makes the results worse, however the method is immune to light smoothing.

7. Conclusions

Different methods for labeling digital images are investigated. The methods, that add the label in the spatial domain, seem to have the lowest bit capacity and the lowest resistance to JPEG compression. Adding the label in another domain sometimes improves the bit capacity and the resistance. The use of the DCT transform gives the best results, obviously because the JPEG algorithm makes use of the same transform. The resistance can be increased further if the quantization step is also taken into account. If the original unlabeled image can be used together with the labeled one to check the label, the capacity and the resistance to JPEG compression seem to be higher.
Only a few existing labeling techniques are suitable for a copy protection system. However, from these

methods only one DCT based method has a sufficient bit capacity and an acceptable resistance to JPEG compression. Therefore, the spatial labeling methods are developed further to achieve the same results. By allowing smaller blocks to embed one label bit, making the embedding level dependent on a lower quality JPEG compressed version of the image and adapting the random pattern, this aim is reached. Using the extended method some true color images were labeled with a few hundred bits. The label turned out to be resistant to JPEG compression, with quality parameter set to 40% (compression rate >1:20).
This method can be improved further by rejecting blocks if the embedding level becomes to high.
A disadvantage of almost all methods mentioned in this paper including the extended one, is that they are not resistant to rotations, cropping, translations and scaling. This problem could maybe be solved by taking into account contour information to find one or two orientation points in the image.

References

[1] Digital Audio Interface, International Standard IEC 958

[2] RACE M 1005: Access control and copyright protection for images (ACCOPI), Workpackage 5, June, 1995

[3] TALISMAN: http://www.tele.ucl.ac.be/IMAGES/ACTS/talisman.html

[4] R.G. van Schyndel, A.Z. Tirkel, C.F. Osborne : "A Digital Watermark", Int. Conf. on Image Processing, volume 2, pages 86-90, IEEE, 1994

[5] W. Bender, D. Gruhl, N. Morimoto : "Techniques for Data Hiding", Proceedings of the SPIE, 2420:40, San Jose CA, USA, February 1995

[6] I. Pitas, T. Kaskalis : "Signature Casting on Digital Images", Proceedings IEEE Workshop on Nonlinear Signal and Image Processing, Neos Marmaras, June, 1995

[7] Caronni G.: "Assuring Ownership Rights for Digital Images", Proceedings of Reliable IT Systems, VIS '95, Vieweg Publishing Company, Germany, 1995

[8] J. Zhao, E. Koch : "Embedding Robust Labels into Images for Copyright Protection", Proceedings of the International Congress on Intellectual Property Rights for Specialized Information, Knowledge and New Technologies, Vienna, Austria, August 1995

[9] E. Koch, J. Zhao : "Towards Robust and Hidden Image Copyright Labeling", Proceedings IEEE Workshop on Nonlinear Signal and Image Processing, Neos Marmaras, June, 1995

[10] I.J. Cox, J. Kilian, T. Leighton, T. Shamoon : "Secure Spread Spectrum Watermarking for Multimedia", NEC Research Institute, Technical Report 95 - 10

[11] F.M. Boland, J.J.K. O Ruanaidh, C. Dautzenberg : "Watermarking Digital Images for Copyright Protection", Proceedings of the 5th International Conference on Image Processing and its Applications, no 410, Edinburgh, July, 1995

Back to my home page

TU Delft, IT Group, Dept. Elec. Eng., P.O. Box 5031, Delft, The Netherlands **Phone: (+31)15-278 3084**
E-mail: gerhard@it.et.tudelft.nl
URL: http://www-it.et.tudelft.nl/people/gerhard/home.html
Last modified: July 17 1996

17

PROCEEDINGS
EUROPTO
SERIES

Digital Compression Technologies and Systems for Video Communications

Naohisa Ohta
Chair/Editor

7–9 October 1996
Berlin, FRG

Sponsored by
Technologiestiftung Innovationszentrum Berlin eV
IS&T—The Society for Imaging Science and Technology
EOS—The European Optical Society
The Commission of the European Communities, Directorate General for Science, Research, and Development

Published by
SPIE—The International Society for Optical Engineering

SPIE
P
PROCEEDINGS SERIES

Volume 2952

SPIE is an international technical society dedicated to advancing engineering and scientific applications of optical, photonic, imaging, electronic, and optoelectronic technologies.

Digital Watermarking of Raw and Compressed Video

Frank Hartung

Bernd Girod

Telecommunications Institute
University of Erlangen-Nuremberg
Cauerstrasse 7, 91058 Erlangen, Germany
{hartung, girod}@nt.e-technik.uni-erlangen.de

ABSTRACT

Embedding information into multimedia data is a topic that has gained increasing attention recently. For video broadcast applications, watermarking of video, and especially of already encoded video, is interesting. We present a scheme for robust interoperable watermarking of MPEG-2 encoded video. The watermark is embedded either into the uncoded video or into the MPEG-2 bitstream, and can be retrieved from the decoded video. The scheme working on encoded video is of much lower complexity than a complete decoding process followed by watermarking in the pixel domain and re-encoding. Although an existing MPEG-2 bitstream is partly altered, the scheme avoids drift problems. The scheme has been implemented and practical results show that a robust watermark can be embedded into MPEG encoded video which can be used to transmit arbitrary binary information at a data rate of several bytes/second.

Keywords: watermarking, video, MPEG-2, video broadcast

1 Introduction

With digital broadcast of video, legal issues of copyright protection have become more important, since the inherent decrease of quality of analog video duplication has vanished in digital applications. A favorable method of copyright protection is digital watermarking of the multimedia data, i.e., adding a "watermark" (in other publications also called "label", "tag" or "signature") that authenticates the legal copyright holder and that cannot be manipulated or removed without, at the same time, impairing the multimedia data so much that they are of no commercial value any more.[1-8] Alternatively, an individual watermark might also be included at the conditional access unit in the transmitter that encrypts the video for the individual receiver in order to identify the receiver if he copies and illegally distributes the video, as shown in Fig. 1. While previous publications[1-8] do not deal with watermarking in the bitstream domain of coded video, this is an especially interesting topic. High-quality MPEG encoding is very complex, in some applications it is even done interactively with fine-tuning of parameters by a human operator. Therefore, individual digital watermarking of digital video broadcasted to different receivers can be done only after encoding, but before decoding, as shown in Fig. 1.

0-8194-2356-4/96/$6.00

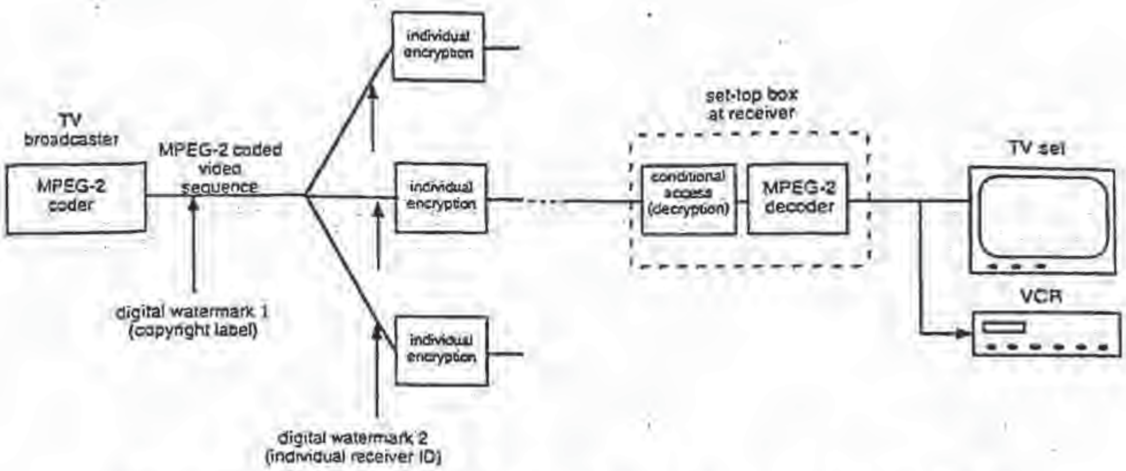


Figure 1: Transmission scheme for video with individual watermark embedding.

In section 3, we introduce a scheme for spread spectrum like watermarking of uncoded video. In section 4, we show techniques for robust *interoperable* digital watermarking of video where we incorporate the watermark in the bitstream domain of MPEG-2 coded video (that is, without decoding and full re-encoding) and can retrieve it from the decoded video, as shown in Fig. 2. In section 5, possible attacks against watermarks are explained, and

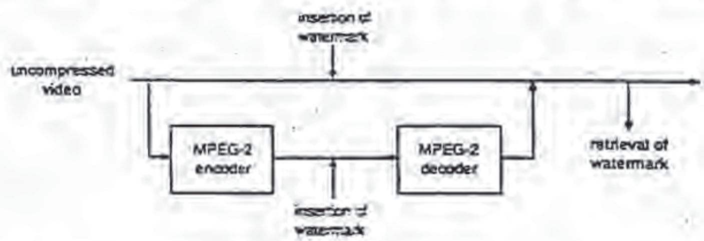


Figure 2: Interoperability of watermarking in the uncoded and coded domain.

remedies are given. In section 6, we present practical results. We have implemented our scheme for watermarking of MPEG-2 encoded video which works robust and can embed arbitrary watermark information into encoded video at a data-rate of several bytes/second.

2 Requirements on a digital watermarking scheme for video broadcast applications

A digital watermark is a signal carrying information that is embedded into another transport signal, for example into a video signal. A watermarking scheme for video broadcast applications should comply with the following requirements:

- The digital watermark embedded into the video data should be invisible or at least hardly perceptible.

- The watermark should be such that it cannot be removed by intentional or unintentional operations on the bitstream or on the decoded video without, at the same time, degrading the perceived quality of the video so much that it is of no commercial value any more. This requirement is called robustness.
- For broadcast applications, it can be assumed that the broadcaster will usually store the video in compressed format. Therefore, it must be possible to incorporate the watermark into the encoded video, i.e., into the bitstream. It is not feasible to decode and re-encode the video for the purpose of watermarking it.
- Watermarking in the bitstream domain may not increase the bit-rate (at least for constant bit-rate applications). This requirement is not obeyed by previous publications dealing with watermarking of still images during JPEG compression.[2]
- It can be assumed (and it is, in practice, the case) that incorporating a watermark into compressed video has to obey much more constraints than incorporating a watermark into uncompressed video. Therefore, it is advantageous to do so in the domain of uncompressed video wherever possible. Hence, the watermarking algorithm should work interoperable for compressed and uncompressed video with the same type of decoder, that is, watermark detector (see Fig. 2).

3 Digital Watermarking of Raw Video

The basic idea of watermarking for video is addition of a pseudo-random signal to the video that is below the threshold of perception and that cannot be identified and thus removed without knowledge of the parameters of the watermarking algorithm.

Our approach to accomplish this is a direct extension of ideas from direct-sequence spread spectrum communications.[8] The approach in[3] is similar and was developed independently. Let us denote

$$a_j, \quad a_j \in \{-1, 1\} \tag{1}$$

a sequence of information bits we want to hide in the video stream. We then spread this discrete signal by a large factor cr, called the chip-rate, and obtain the spread sequence

$$b_i = a_j, \quad j \cdot cr \le i < (j+1) \cdot cr \tag{2}$$

The spread sequence b_i is amplified with an amplitude factor α and modulated with a binary pseudo-noise sequence

$$p_i, \quad p_i \in \{-1, 1\} \tag{3}$$

The modulated signal, i.e. the watermark $w_i = \alpha \cdot b_i \cdot p_i$ is added to the line-scanned digital video signal v_i yielding a watermarked video signal

$$\tilde{v}_i = v_i + \alpha \cdot b_i \cdot p_i. \tag{4}$$

Due to the noisy nature of p_i, w_i is also a noise-like signal and thus difficult to detect, locate, and manipulate. The recovery of the hidden information is easily accomplished by multiplying the watermarked video signal with the same pseudo-noise sequence p_i that was used in the coder:

$$s_j = \sum_{i=j \cdot cr}^{(j+1)\cdot cr-1} p_i \cdot \tilde{v}_i = \sum_{i=j \cdot cr}^{(j+1)\cdot cr-1} p_i \cdot v_i + \sum_{i=j \cdot cr}^{(j+1)\cdot cr-1} p_i^2 \cdot \alpha \cdot b_i \tag{5}$$

The first term on the right-hand side of (5) vanishes, if

$$\sum_{i=j \cdot cr}^{(j+1)\cdot cr-1} p_i = 0 \tag{6}$$

(i.e., the pseudo-noise sequence contains as many -1's as 1's in the interval $[j \cdot cr \ldots (j+1) \cdot cr)$), p_i and v_i are uncorrelated and therefore $\sum_{i=j\cdot cr}^{(j+1)\cdot cr-1} p_i \cdot v_i = 0$. In practice however, the sum in (6) is not zero, and a correction term

$$\Delta = -\left(\sum_{i=j\cdot cr}^{(j+1)\cdot cr-1} p_i\right) \cdot mean(\tilde{v}_i), \tag{7}$$

which accounts for the different number of -1's and 1's in the pseudo-noise sequence, has to be added. s_j then ideally becomes

$$s_j = \sum_{i=j\cdot cr}^{(j+1)\cdot cr-1} p_i \cdot \tilde{v}_i + \Delta \approx cr \cdot \alpha \cdot a_j \tag{8}$$

and the recovered information bit $\hat{a}_j$ is

$$\hat{a}_j = sign(s_j). \tag{9}$$

A condition for the scheme to work is that for demodulation the same pseudo-noise sequence p_i is used that was used for modulation. Thus, even if the receiver knows the basic scheme, it cannot recover the information without knowledge of the pseudo-noise sequence and its possible shift. For simplicity, we have assumed a binary pseudo-noise sequence in (3). Non-binary PN sequences are also possible without modifications of the scheme, and are in fact favorable in terms of security. Given several sequences with different watermarks, it is easier to figure out the unwatermarked pixel values if the watermark consists only of -1's and 1's. The amplitude factor α can be varied according to local properties of the image and can be used to exploit spatial and temporal masking effects of the human visual system (HVS). Also, an error correcting code can be employed to increase the robustness of the scheme. Several watermarks can be superimposed, if different pseudo-noise sequences are used for modulation. This is due to the fact that different pseudo-noise sequences are in general orthogonal to each other and do not significantly interfere.[9]

4 Digital Watermarking of Compressed Video

In the bitstream domain it is more difficult to embed a watermark into video, especially when the requirement is imposed that the bit-rate may not be increased. MPEG-2 bitstream syntax allows for user data being incorporated into the bitstream (field user_data, can be included in any of sequence, group_of_pictures and picture headers). However, this is not a suitable means of embedding a watermark, since the user data can easily be stripped off the bitstream. Also, adding user data to an MPEG-2 encoded video sequence increases the bit-rate. Again the key idea is to incorporate the watermark into the signal itself, i.e., into the bitstream representing the video frames. In order to understand how we can achieve that we have to take a close look on how a signal block corresponds to the equivalent portion of the bitstream. Let us consider a block of 8×8 samples, originating from a frame of the sequence for I-frames or from a prediction error signal for P- and B-frames, respectively. The block is transformed with the DCT, quantized, zig-zag-scanned and run-level-encoded with VLC codewords for the (run,level)-pairs. Thus, the block of 8×8 samples translates into a codeword representing the DC coefficient followed by a number of VLC codewords representing (run,level)-pairs and hence specifying position and value of one DCT coefficient each. The (run,level)-codewords in MPEG-2 are fixed. Fig. 3 shows the number of bits for the (run,level)-codewords specified in the MPEG-2 VLC tables.[10] (run,level)-combinations that are not specifically represented in the VLC tables are coded with a codeword of 24 bits. In order to add a watermark, we process the encoded video signal block by block. For each signal block, the watermarking procedure consists of the following steps:

1. Calculate the DCT of the watermark (of the spread information bits modulated by the pseudo-noise sequence) for the 8×8-block. Do a zig-zag-scan, yielding a 1×64-vector of re-scanned DCT coefficients. Denote the DCT coefficients by W_n with W_0 being the DC coefficient and W_{63} being the highest-frequent

VLC codeword lengths for (run,level)-combinations

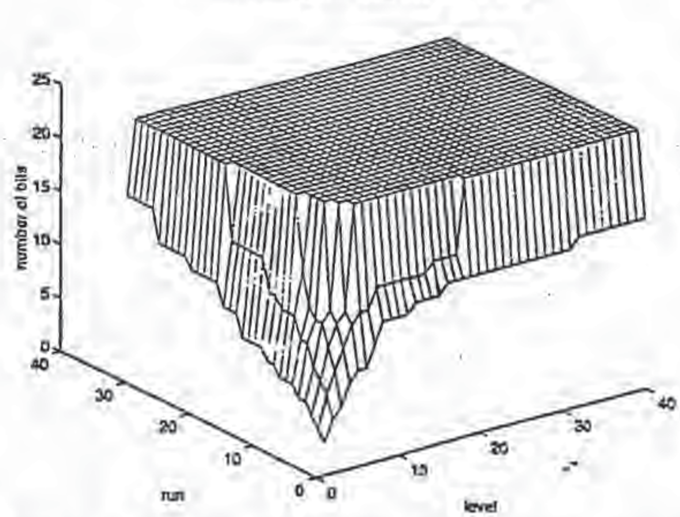


Figure 3: MPEG-2 VLC codeword lengths for (run,level)-codewords.

AC-coefficient. Denote the DCT coefficients of the unwatermarked signal V_n and of the watermarked signal $\hat{V}_n$.

2. **DC-coefficient:** For the DC-coefficient, $\hat{V}_0 = V_0 + W_0$, that is, the mean value of the watermark-block is added to the mean value of the signal-block.

3. **AC-coefficients:** Search the bitstream of the coded signal for the next VLC codeword, identify the (run,level)-pair (r_m, l_m) belonging to that codeword and, thus, the position and amplitude of the AC DCT coefficient V_m represented by the VLC codeword.

4. $\hat{V}_m = V_m + W_m$ is the candidate DCT coefficient for the watermarked signal. However, we do also have the constraint of not increasing the bit-rate. Thus, we have to check the number of bits we have to transmit for the watermarked DCT coefficient $\hat{V}_m$ versus the bit-rate we have to transmit for the unwatermarked DCT coefficient V_m:

5. Let R be the number of bits used for transmitting the codeword for (r_m, l_m) (i.e., for V_m) and $\hat{R}$ be the number of bits used for transmitting the codeword for $(r_m, \hat{l}_m)$ (i.e., for $\hat{V}_m$). (R and $\hat{R}$ are determined by the VLC-tables defined in MPEG-2[10]).

6. If the bit-rate shall not be increased and $R \geq \hat{R}$ (or if the bit-rate of the video may be increased, unconditionally), transmit the codeword for $(r_m, \hat{l}_m)$. Else, transmit the codeword for (r_m, l_m).

7. Repeat steps 3 to 6 until an end-of-block (EOB) codeword is encountered.

Due to the bit-rate constraint, usually only few DCT coefficients of the watermark can be incorporated per 8×8-block, in a lot of cases (especially for coarse quantization) it might be only the DC coefficient as outlined in step 2. As a result, the watermarking scheme in the bitstream domain is less robust than its counterpart in the pixel domain. In other words: in the bitstream domain, only a fraction of the signal energy of the watermark can successfully be embedded. However, for watermarking of video, the chip-rate cr may be chosen to be very high, increasing the robustness to the desired level, but at the same time decreasing the data rate for the watermark.

In practice, step 4 has to be modified in order to avoid drift, which otherwise might occur because we partly alter a previously encoded bitstream. We have to decode the unwatermarked video in parallel and to add not only the watermark, but also to subtract the drift that has been occurred so far.

5 Attacks against Watermarks, and Remedies

One of the main requirements on watermarking schemes is robustness against intentional or unintentional attacks attempting to remove or destroy the watermark. Possible attacks include:

a. Addition of a constant offset

b. Addition of gaussian or non-gaussian noise

c. Linear filtering, e.g. low-pass or high-pass filtering

d. Nonlinear filtering, e.g. median filtering

e. Compression, e.g. by hybrid coding schemes like MPEG or H.263

f. Local exchange of pixels (e.g. permutation of a 2×2-block of pixels)

g. Quantization of the pixel gray values

h. Rotation of the video frames

i. Spatial scaling of the video frames

j. Removal or insertion of single pixels

k. Removal or insertion of pixel rows or columns

l. Removal or insertion of video frames

m. Averaging of several versions of the same video with different embedded watermarks

n. Single or multiple analog recording on a VCR

The attacks listed in a.– g. do not pose a real problem to our scheme, if the parameters (especially the chip-rate) are chosen adequately. The same holds for rotation of the video frames (h.), if the rotation angle is very small; otherwise a rotation detection and correction has to be added. Spatial scaling (i.) is critical and a scaling detection and correction mechanism is needed. Removal or insertion of parts of the data (j.–l.) leads to loss of synchronicity of the PN sequence between sender and receiver, and must be considered. A scheme that detects loss of synchronicity and attempts to resynchronize (for example by use of a sliding correlator[9]) must be employed. If complexity has not to be considered, all mentioned attacks can be counter-attacked. A real problem however occurs if several versions of the same video with different embedded watermarks are averaged in order to reconstruct the original pixel values (m.). Countermeasures against this sort of attack are still under research. The effects of analog recording (n.) are typically a combination of the effects mentioned before.

6 Implementation and Simulation Results

We have implemented the outlined scheme as a C program which takes an MPEG-2 bitstream as its input. The program decodes the video and simultaneously parses the bitstream and writes it to a new file. Only those parts of the bitstream containing VLC codewords representing DC- and AC-coefficients of DCT blocks are located and replaced by VLC codewords representing DC- and AC-coefficients of the same block *plus watermark*. Typical parameters are $\alpha = 1 \ldots 5$ and $c = 10,000 \ldots 1,000,000$, yielding data rates for the watermark of $1.25 \ldots 125$ bytes/second for NTSC TV resolution. The complexity, as shown in Fig. 4, is much lower than the complexity of a

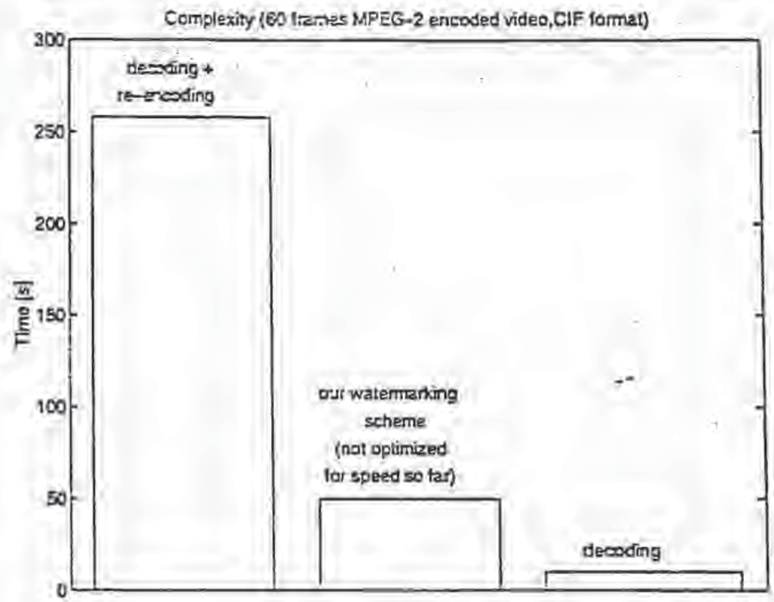


Figure 4: Complexity of our watermarking scheme compared to encoding and decoding.

decoding process followed by watermarking in the pixel domain and re-encoding. For comparison, the complexity of decoding alone is also given. Please note that our program, unlike the public domain MPEG coder and decoder, has not been optimized for speed yet. Figures 5-7 show an example frame from a video sequence. Fig. 5 shows

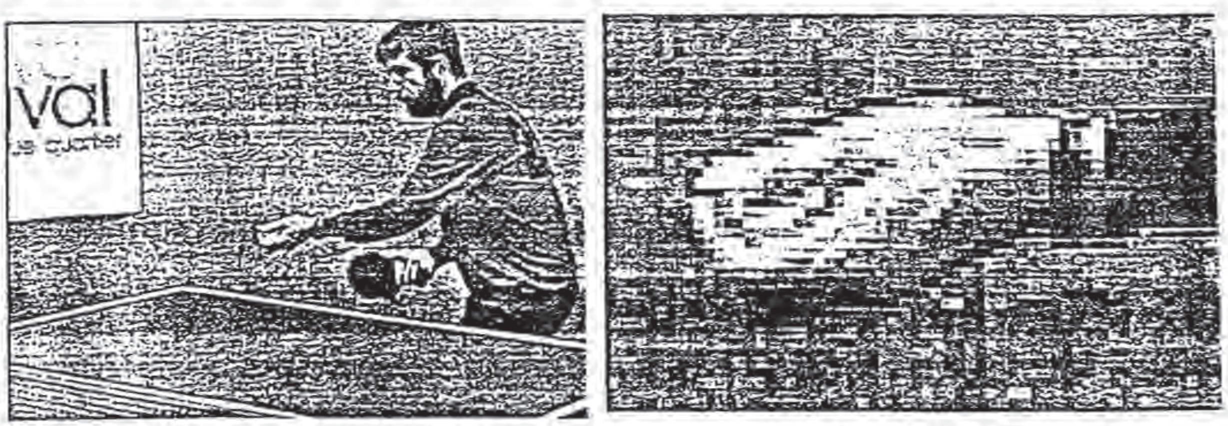


Figure 5: Original

the original frame without compression and a detail from the hand of the table tennis player. Fig. 6 shows the same frame after MPEG-2 encoding and decoding and without an embedded watermark. Fig. 7 finally shows the compressed frame with an embedded watermark. As can be seen, the watermark results in slightly changed pixel amplitudes which are however not visible except in direct comparison to the unwatermarked image. The

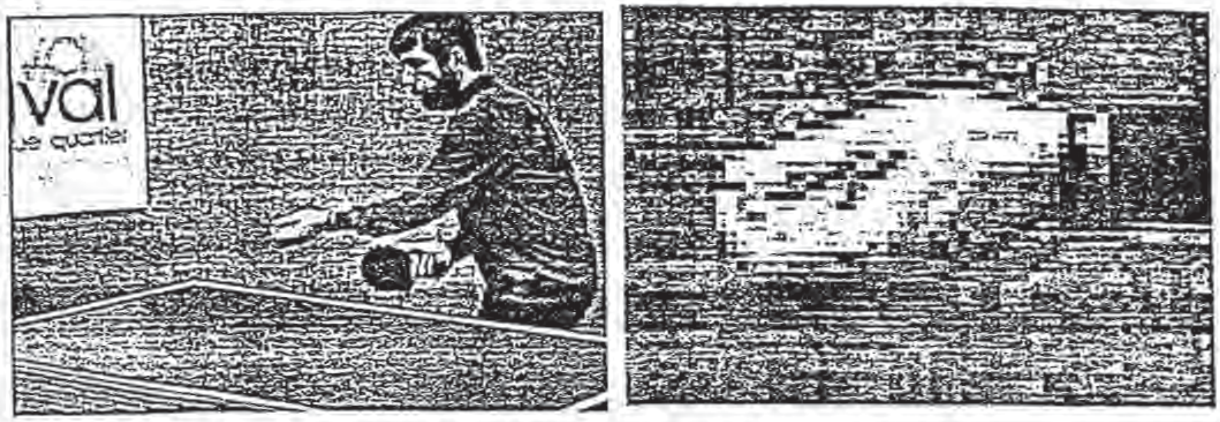


Figure 6: MPEG-2 coded, without watermark

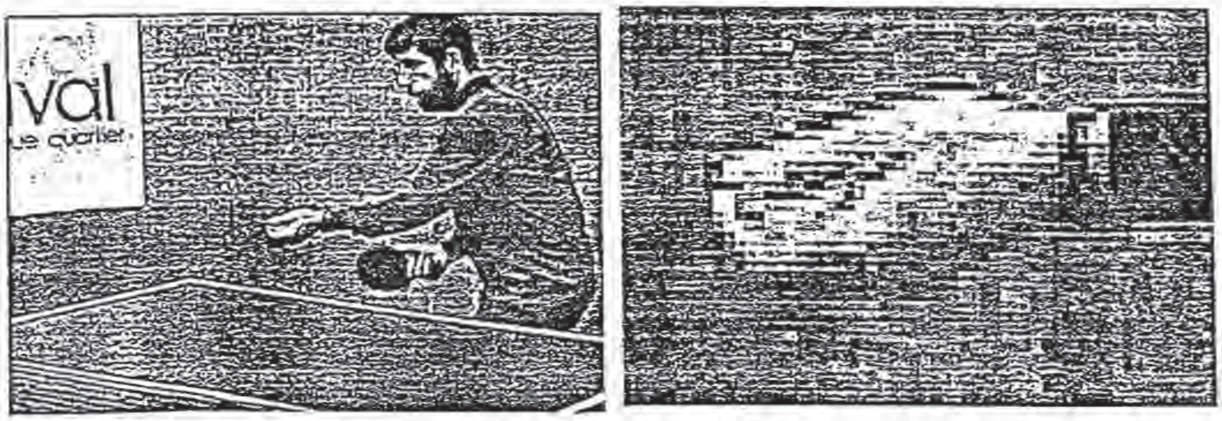


Figure 7: MPEG-2 coded, with embedded watermark

degradation can directly be influenced by varying the amplitude of the watermark. A higher amplitude leads to better robustness, but possibly results in visually annoying distortions.

7 Conclusions

We have presented a novel scheme for watermarking of MPEG-2 compressed video in the bitstream domain. Working on encoded rather than on unencoded video is important for practical watermarking applications. The scheme is interoperable and fully compatible with a scheme working in the pixel domain of uncompressed video which was also presented. With appropriate parameters, the watermarking scheme in the MPEG-2 bitstream domain can achieve netto data rates of several bytes/second while being very robust against unattempted and attempted attacks. The principle can also be applied to other hybrid coding schemes like MPEG-1, ITU-T H.261 or ITU-T H.263.

8 REFERENCES

[1] E. Koch and J. Zhao. Digital copyright labeling: providing evidence of misuse and tracking unauthorized distribution of materials. *OASIS magazine*, December 1995.

[2] E. Koch and J. Zhao. Towards robust and hidden image copyright labeling. In *Proceedings of 1995 IEEE Workshop on Nonlinear Signal and Image processing*, Neos Marmaras, Greece, June 1995.

[3] I. Cox, J. Kilian, T. Leighton, and T. Shamoon. Secure spread spectrum watermarking for multimedia. Technical Report 95-10, NEC Research Institute, Princeton, NJ, USA, 1995.

[4] N. Nikolaidis and I. Pitas. Copyright protection of images using robust digital signatures. In *Proceedings ICASSP 96*, May 1996.

[5] Germano Caronni. Ermitteln unauthorisierter Verteiler von maschinenlesbaren Daten. Technical report, ETH Zürich, Switzerland, August 1993.

[6] Germano Caronni. Assuring ownership rights for digital images. In *Proceedings VIS 95, Session "Reliable IT Systems"*. Vieweg, 1995.

[7] Walter Bender, Daniel Gruhl, and Norishige Morimoto. Techniques for data hiding. Technical report, MIT Media Lab, 1996.

[8] ACCOPI. RACE project M1005 (ACCOPI): Workpackage 8: Watermarking techniques. Technical report, ACCOPI Consortium, April 1995.

[9] David L. Nicholson. *Spread Spectrum Signal Design - Low Probability of Exploitation and Anti-Jam Systems*. Computer Science Press, 1988.

[10] ISO/IEC 13818-2, Generic Coding of Moving Pictures and Associated Audio, Recommendation H.262 (MPEG-2), 1995. International Standard.

RC 20509 (July 25, 1996)
Computer Science/Mathematics

IBM Research Report

Can Invisible Watermarks Resolve Rightful Ownerships?

Scott Craver

Department of Mathematics
Northern Illinois University
DeKalb, IL 60115
Email: caj@niu.edu

Nasir Memon

Department of Computer Science
Northern Illinois University
DeKalb, IL 60115
Email: memon@cs.niu.edu

Boon-Lock Yeo

IBM Research Division
T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598
Email: yeo@watson.ibm.com

Minerva Yeung

Department of Electrical Engineering
Princeton University
Princeton, NJ 08544
Email: mingy@ee.princeton.edu

IBM Research Division
Almaden · T.J. Watson · Tokyo · Zurich

Can Invisible Watermarks Resolve Rightful Ownerships?

Abstract

Digital watermarks have been proposed in recent literature as the means for copyright protection of multimedia data. In this paper we address the capability of invisible watermarking schemes to resolve copyright ownerships. We will show that rightful ownerships cannot be resolved by current watermarking schemes alone. In addition, in the absence of standardization of watermarking procedures, anyone can claim ownership of any watermarked image. Specifically, we provide counterfeit watermarking schemes that can be performed on a watermarked image to allow multiple claims of rightful ownerships. We also propose *non-invertible* watermarking schemes in this paper and discuss in general the usefulness of digital watermarks in identifying the rightful copyright owners. The results, coupled with the recent attacks on some image watermarks, further imply that we have to carefully re-think our approaches to invisible watermarking of images, and re-evaluate the promises, applications and limitations of such digital means of copyright protection.

Keywords: Authentication of copyright ownerships, invisible watermarks, copyright protection of images, attacks on watermarking schemes, non-invertible watermarking techniques.

1 Introduction

The rapid growth of digital imagery has called upon the needs for effective copyright protection tools. Various watermarking schemes and software products have been introduced recently in an attempt to address this growing concern. It is natural to ask a few questions regarding all these efforts: (1) What is a digital watermark? (2) Why are digital watermarks necessary, or in other words, what can digital watermarks achieve, or fail to achieve? (3) How useful are digital watermarks, or in other words, what can digital watermarks do for copyright protection in addition to current copyright laws, or current avenues of resolving copyright grievances?

In general, there are two types of digital watermarks (signatures) addressed in existing literature: visible and invisible watermarks[1]. These watermarks are developed mainly for two purposes: copyright protection and data authentication. In this paper we shall focus on the large class of invisible watermarks developed for one instance of copyright protection — that is, to identify the rightful owner. In this case the ownership labels which are embedded in an image have to be recoverable despite intentional or unintentional modification of the image. This means that such labels (and thus the corresponding labeling techniques) should ideally be robust against normal image processing operations like filtering, requantization, dithering, scaling, cropping, etc. and common image compression like JPEG image compression standards. They must also be invulnerable to deliberate attempts to forge, remove or invalidate labels.

Existing invisible watermarking schemes for copyright protection have been reported in research literature (for example, see [1, 2, 3, 4]). Unfortunately, many of these schemes did not address the *ends* of invisible watermarking schemes. They instead focused on the robust *means* to mark an image invisibly. In doing so, the concerns brought up by the previous three questions may not be properly and clearly addressed. We will show in this paper that current invisible watermarking schemes cannot resolve rightful ownership of any image watermarked with multiple signatures (labels). In addition, without any standardization of watermarking techniques or specification of certain requirements in the watermarking procedures (that is, without properly answering the question "What is a

[1]Some papers, such as [1], discuss watermarking other forms of multimedia data such as sound clips. Our research has focussed on image data, and hence we say "invisible" when in a wider sense we mean "imperceptible." The idea presented in this paper applies also to other form of multimedia data.

digital watermark?"), we shall show that anyone can claim ownership of an image by simple methods described in later sections. The results, coupled with the recent attacks on some of the image watermarks reported in [5], further suggest that we have to carefully re-think our approaches to invisible watermarking of images, and re-evaluate the promises of such digital means of copyright protection. In other words, it is crucial that any watermarking scheme proposed for copyright protection be able to answer the last two questions: "Why is it necessary?" and "How useful is it?".

The paper is organized as follows: we shall give the general definitions and formulations of digital watermarking schemes in Section 2. In Section 3, we discuss how digital watermarking can be used to resolve rightful ownerships, and depict a scenario in which there may be more than one "rightful" owners of an image. We then show in Section 4 that such a scenario can actually be created by developing counterfeit watermarking schemes that can be performed on a watermarked image to allow multiple claims of rightful ownerships. An implementation of such a scheme, which is used to invalidate the watermarking method proposed by [1] is also described. In Section 5 we present the *non-invertible* watermarking schemes as a method of preventing the type of attack described in Section 4. We conclude in Section 6 with a discussion on the use of watermarking schemes for the authentication of rightful ownerships.

At this point we would like to emphasize that we still believe invisible watermarks are important to the information infrastructure, with applications that include determining rightful copyright ownerships. However, resolving rightful ownerships of digital images may require, in addition to invisible watermarks, the inclusion of protocols, formal requirements and standardization similar to traditional legal channels that are currently used to copyright images and photographs. Through this paper, we hope to promote new discussions and interests in the research community on the applications and values, as well as limitations, of digital signatures and the corresponding watermarking techniques.

2 Watermarking of Images: Definitions and Formulations

In this section we shall give a generalized formulation of invisible watermarking schemes. We shall define in general terms the process of signature insertion into an image and the use of invisible watermarks to determine the ownership of a watermarked image. Figure

1 illustrates both the encoding process in which a signature is inserted into an image, and the decoding process in which a signature is recovered and then compared to the inserted signature.

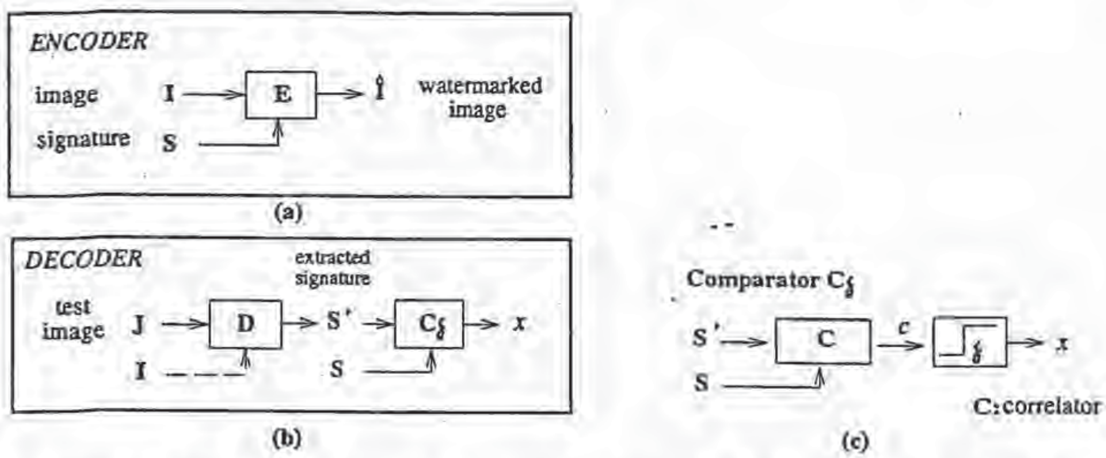


Figure 1: Encoding and Decoding embedded signatures in an image: (a) encoder (b) decoder (c) comparator

Here we denote an image by I, a signature $S = \{s_1, s_2, \cdots\}$ and the watermarked image by $\hat{I}$. $\mathcal{E}$ is an encoder function if it takes an image I and a signature S, and generates a new image which is called the *watermarked* image $\hat{I}$, i.e.,

$$\mathcal{E}(I, S) = \hat{I}. \tag{1}$$

It should be noted that we do not exclude the possibility that the signature S is dependent on the image I. In such cases, the encoding process described by (1) still holds. A diagram of the encoding process is shown in Figure 1(a).

A decoder function $\mathcal{D}$ takes an image J (J can be a watermarked or un-watermarked image, and possibly corrupted) whose ownership is to be determined, and recovers a signature S' from the image. In this process, an additional image I can also be included which is often the original (and un-watermarked) version of J. This is due to the fact that some encoding schemes may make use of the original images in the watermarking process to provide extra robustness against intentional and unintentional corruption of pixels.

$$\mathcal{D}(J, I) = S'. \tag{2}$$

The extracted signature S' will then compared with the owner signature sequence by a comparator function $\mathcal{C}_\delta$, and a binary output decision is generated. It is a 1 if there is a match and 0 otherwise:

$$\mathcal{C}_\delta(S', S) = \begin{cases} 1, & c \geq \delta; \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Here, c is the correlation of the two signatures. A diagram of the decoding process is shown in Figure 1(b), and the comparator is depicted in Figure 1(c). Without loss of generality, watermarking schemes can be treated as a three-tuple $(\mathcal{E}, \mathcal{D}, \mathcal{C}_\delta)$.

The above framework describes what an invisible watermark is and how it can potentially be used to determine ownership. This is a generalized formulation. It does not give any insight into how exactly a watermarking scheme works. In view of this, here we specify some watermarking schemes. In particular, we describe the formulation of a class of invisible watermarking schemes, which we call *feature-based* watermarking schemes, that embed a signature $S = \{s_1, s_2, \cdots\}$ into some set of derived *features* $D(I) = \{f_1(I), f_2(I), \cdots\}$. The embedding process is achieved by an *insertion operation* which we denote by the symbol $\oplus$. That is,

$$f'_i = f_i \oplus s_i.$$

The insertion operation has an inverse operation, namely the *extraction operation*, which we denote by $\ominus$. That is,

$$f'_i \ominus f_i = s_i.$$

Note that, for notational simplicity we take the insertion (and extraction) process to be binary operators, although in general they could be arbitrary functions of f_i and s_i.

Usually, the feature set $\{f_1(I), f_2(I), \cdots\}$ is chosen such that slight modification of individual features does not *perceptually* degrade image I. In addition it is also desirable that each element in this set of features will not be changed significantly when the image is not perceptually degraded. An example of such a set features would be transformed domain (e.g., DCT, wavelet) coefficients which contain significant energy content. The labels s_i that compose the watermark in this case could be real numbers drawn from a specific distribution and the insertion operation could simply be the addition of s_i to these coefficients.

Example 1 *An invisible watermarking scheme as proposed by Cox* et al. *[1].*

In this scheme, the 2D DCT is taken of the image I and the set $D(I)$ corresponds to the n AC DCT coefficients of highest magnitude. Such coefficients will typically correspond to low frequency ones. Significant modification made to the image will cause image fidelity to degrade before the watermark does.

The encoder $\mathcal{E}$ takes a signature S and places it in the set $D(I)$. An inverse 2D DCT is taken of this modified matrix, yielding the watermarked image $\hat{I}$. To determine if a given image J contains the signature S, the decoder $\mathcal{D}$ first extract $T = \{t_1, t_2, \cdots\}$ from J as follows:

$$t_i = f_i(J) - f_i(I). \tag{4}$$

The confidence measure c is then taken to be the

$$c = \frac{\sum_i t_i.s_i}{\sqrt{\sum_i t_i^2}} \tag{5}$$

Alternatively, the normalized correlation

$$c = \frac{\sum_i t_i.s_i}{\sqrt{(\sum_i t_i^2 \sum_i s_i^2)}} \tag{6}$$

can be used. In this case, if $J = \hat{I}$, then $c = 1$. If J is a modified version of $\hat{I}$, and the changes are not perceptually significant, c will be large value but smaller than 1.

□

Throughout the rest of this paper, we shall use two fictional characters, Alice and Bob, to illustrate the various scenarios involving the claims of copyright ownerships and to bring up the different issues of the application of digital watermarks in resolving rightful ownerships.

3 Resolving Rightful Ownerships by Invisible Watermarks

It is a common view that invisible watermarking schemes may be used to protect the rights of copyright owners of images: at the very least, the labels (in other words, the digital signatures) extracted from the watermarked images can be used to identify the rightful owners. But how can we do this? Does it mean straight-forwardly that the one whose signature matches the embedded signature extracted from an image will automatically be the rightful owner of the image?

> *Suppose Alice and Bob use the same digital watermarking technique to watermark their images. This means that there is one unique decoding scheme to extract the labels embedded in the images. If a label extracted from a watermarked image matches the particular signature label of Alice, then the image is believed to belong to her. Similarly, if the label matches Bob's signature, then it must be his image. If a watermarked image contains both Alice and Bob's signatures, whose image is it?*
>
> *Suppose now that Alice and Bob use different watermarking techniques. Given a watermarked image, Alice can take this image and decode the label using her decoding scheme. Similarly Bob can perform the label extraction process with his decoding scheme. If Alice's decoder indicates that the image belongs to her while Bob's decoder indicates that it is his image, whose image is it?*

The question of how to determine or resolve rightful ownership of an image in the face of multiple copyright ownership claims has never been explicitly raised, or answered. But the scenario is valid, given that an image can be generated and modified digitally, and any image that is watermarked by Alice and in circulation can be watermarked again by Bob. In such cases Alice and Bob can use the same watermarking techniques, or apply different ones.

Of course, somewhere out in the dark, there are the so-called original images (or, *unwatermarked* images). Without proper copyright registration and the traditional protection of copyright laws, (after all, why are digital signatures necessary if copyright laws can fully protect the interests of the copyright owners?) one can always look to these original images for an answer.

Now there is one watermarked image from which the digital signatures of both Alice and Bob have been extracted and both of them are claiming to be its rightful owner. Alice can ask Bob for his original image and check if it contains her signature. Similarly, Bob can ask Alice for her original image and check for his signature. If Bob took Alice's watermarked image and introduced his own watermark into it, then both Bob's "original" and watermarked images contain Alice's mark. Alice's original does not contain Bob's.

Thus, by keeping her original image locked away with the details of the watermark label, Alice can easily foil any such ex post facto *watermarking of her image. This is because Bob does not have the access to Alice's original image, even if he has access to the watermarked version of the image.*

Or can she? If Alice's original contains Bob's signature and vice versa, *who owns this image: Alice or Bob?*

In such a case rightful ownership cannot be resolved by invisible watermarks alone. We shall show in the following section that this scenario is not hypothetical, but can be engineered with current watermarking schemes. We will present in detail a counterfeit watermarking scheme that allows multiple claims of ownerships. More precisely, the true owner of an image can no longer argue his or her claim based only on the digital signatures that invisibly embedded in it, as others can engineer an equal amount of evidence that they too own the image. The situation will not happen with visible watermarks such as the one proposed in [6].

4 Invalidating Claims of Ownerships

To invalidate claims of ownerships of an image, it is necessary to generate the confusion illustrated in the case of Alice and Bob — that there are two original images, each contains the watermark of the other party. But in reality there is one and only one original. We shall show in this section how to create another "original" image $\hat{I}'$ (the counterfeit original) from a watermarked image $\hat{I}$, without the access to the true original image I. More formally, given $\hat{I}$ which is watermarked by a watermarking scheme $(\mathcal{E}, \mathcal{D}, C_\delta)$, we have in procession $\hat{I}'$, S' and a decoding function $\mathcal{D}'$ such that the following properties are satisfied:

1. $C_{\delta}(\mathcal{D}(\hat{I}', I), S) = 1$.

2. $C_{\delta'}(\mathcal{D}'(I, \hat{I}'), S') = 1$.

δ' and δ are sufficiently large thresholds. $\mathcal{D}'$ can be the same as, or different from, the decoding function $\mathcal{D}$. The two parties can use different watermarking schemes in the latter.

> *Alice has an image I. She watermarks it with her watermarking scheme to generate a watermarked image $\hat{I}$ which is then made accessible to the public. Bob takes this watermarked image, and creates a counterfeit original $\hat{I}'$ using our scheme which he then claims to be his original. The first property states that Bob's fabricated "original" $\hat{I}'$ contains Alice's signature S. This is to be expected if the watermarking technique employed by Alice is robust. However, the second property implies that Alice's original image I contains Bob's signature S'!*
>
> *Bob can claim by virtue of property (2) that the image I (Alice's original) actually contains his watermark S'. Of course, Alice, by virtue of property (1), also claims that Bob's "original", $\hat{I}'$, contains her watermark S. Thus, it is not possible to determine the rightful owner of the image.*

Given only $\hat{I}$, we want to construct $C_{\delta'}, \mathcal{D}', \hat{I}'$ and S' such that both properties (1) and (2) are satisfied. This is achieved by *removing* a randomly selected watermark S' instead of embedding the watermark in. The process is illustrated as follows.

> *Bob identifies some features already present in the watermarked image, develops a scheme that removes these features which in return become his signature(s) S', and creates a fake original image $\hat{I}'$ which he then locks away as his "original" along with S', in the same way that Alice would lock away her original I and S. The scenario is shown in Figure 2.*

More precisely, in context of the feature based watermarking schemes described in section 2, the attacker (in this case, Bob) constructs his counterfeit "original" image as follows. He extracts a chosen (possibly random) watermark S' from the set $D'(\hat{I}) = \{f'_i(\hat{I})\}$ to generate an image $\hat{I}'$ such that

$$f'_i(\hat{I}') = f'_i(\hat{I}) \ominus s'_i. \qquad (7)$$

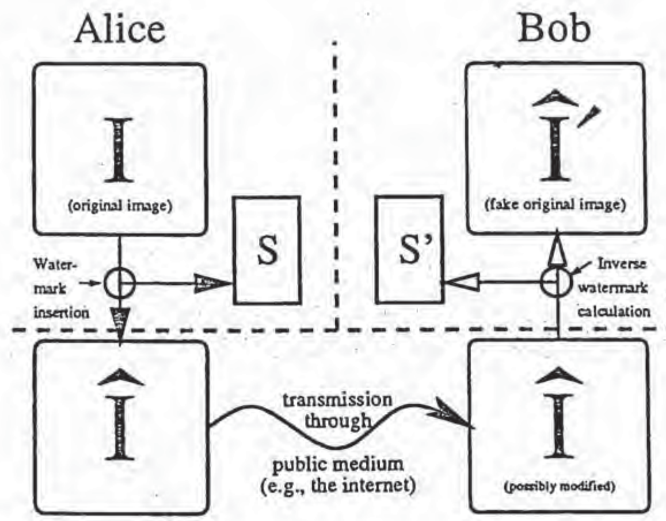


Figure 2: Forging a watermark. Alice watermarks image I to get $\hat{I}$, which she makes public. Bob computes an image $\hat{I}'$ and watermark S', such that watermarking $\hat{I}'$ with S' yields $\hat{I}$.

The set $D'(\hat{I})$ of derived coefficients is assumed to remain more or less the same when the image is not perceptually degraded. The decoding scheme, operating on the counterfeit "original" $\hat{I}'$ and the true original I, first extracts $T' = \{t'_1, t'_2, \cdots\}$ as follows:

$$t'_i = f'_i(I) \ominus f'_i(\hat{I}'). \tag{8}$$

The confidence measure, taken to be the normalized correlation between T and S', defined in (6), is then compared to the threshold δ'. Because of the robustness of the set $D'(\hat{I})$ against perceptually insignificant modification, we can expect that

$$f'_i(I) \approx f'_i(\hat{I}). \tag{9}$$

Combining (7), (8) and (9), we have

$$t'_i \approx s'_i, \tag{10}$$

so that the correlation between T' and S' is large and implies that $C_{\delta'}(T', S')$ will most likely be equal to 1. The attacker (Bob) can thus claim that the true original I contains his signature S' and that I is a modified version of $\hat{I}'$. Conversely, the robustness of watermarking scheme[2] used to embed S onto I allows the true owner (Alice) to also argue that $\hat{I}'$ contains the watermark S. We now have a scenario whereby rightful ownership cannot be resolved through invisible watermarking scheme.

The counterfeiting scheme works by inverting the watermarking process. The key step is (7). By "subtracting off" a watermark S' in $\hat{I}$, we are essentially causing a watermark to be present in I, even when we do not have access to I. Notice that removing an existing watermark is not necessary to create this ownership deadlock, and thus the robustness of a watermark to survive image corruption is not enough to guarantee ownership.

We show an example of how to achieve a counterfeit attack on the class of watermarking schemes whose encoding and decoding process rely on the set of derived coefficients $\{f_1(I), f_2(I), \cdots\}$. In terms of the *insertion* and *extraction* operators, we use simple *addition* $+$ in place of $\oplus$ and *subtraction* in place of $\ominus$.

Example 2 *A detailed analysis when $\mathcal{D} = \mathcal{D}'$, i.e., same decoding scheme and $D() = D'()$, i.e., when the same set of derived coefficients are used in the original watermarking (from I to $\hat{I}$) and the generation of second "original" (from $\hat{I}$ to $\hat{I}'$).*

[2]This is why any watermarking scheme has to be practically robust. Otherwise the attacker, armed with a more robust invisible watermarking scheme, will be able to substantiate his claim of ownership, while the true owner may totally lose his claim because his watermark may be virtually gone after the attack.

Here, we have $f_i = f'_i$. The decoder $\mathcal{D}$, after comparing the original I and the fabricated "original" $\hat{I}'$, extracts $T = \{t_1, t_2, \cdots\}$:

$$\begin{aligned} t_i &= f_i(\hat{I}') - f_i(I) \\ &= f_i(\hat{I}) - s'_i - f_i(I) \\ &= s_i - s'_i \end{aligned}$$

Similarly, the decoder $\mathcal{D}'$, after comparing the fabricated "original" $\hat{I}'$ and the true original I, extracts $T' = \{t'_1, t'_2, \cdots\}$:

$$\begin{aligned} t'_i &= f'_i(I) - f'_i(\hat{I}') \\ &= f_i(I) - f_i(\hat{I}') \\ &= f_i(I) - f_i(\hat{I}) + s'_i \\ &= s'_i - s_i \end{aligned}$$

Thus, $t_i = -t'_i$ and the two correlations are identical:

$$\frac{\sum_i t_i.s_i}{\sqrt{(\sum_i t_i^2 \sum_i s_i^2)}} = \frac{\sum_i t'_i.s'_i}{\sqrt{(\sum_i t_i'^2 \sum_i s_i'^2)}} \tag{11}$$

$$= \frac{\sum_i s_i^2 - \sum_i s_i.s'_i}{\sqrt{(\sum_i (s_i - s'_i)^2 \sum_i s_i^2)}} \tag{12}$$

The second term in (12) is very small if we assume little or no correlation between S and S' (which would be the case in practice) and the whole expression (12) would be large.

□

We have illustrated an extreme case where using the same decoding function and using the same set of derived coefficients actually generate the same correlation values when both parties are trying to establish rightful ownership, which clearly cannot be resolved.

Example 3 *A successful implementation of the proposed attack on the watermarking scheme proposed by Cox et. al. [1].*

In order to provide a more concrete example of counterfeiting an original image we wrote a program to implement the algorithm described in [1]. and then modified it to perform the inverse operation as describe above. We used the same formula that Cox and *et al.* used to insert randomly generated watermark vector elements into an image's 1000 highest AC DCT coefficients v_i, yielding updated coefficients v_i'. To perform the inverse operation of identifying and removing a random watermark, this insertion formula was inverted to compute v_i as a function of v_i', rather than the other way around[3].

This was then unleashed on an already watermarked image $\hat{I}$, using a watermark vector S' to yield a new "original" image $\hat{I}'$ (in reality a fake original) without any visible degradation of image quality. Using (5) as a measure of confidence of a watermark's presence in an image, the fabricated watermark S' is present in the original image I with a confidence value of 23.52, while the original signature S is present in the fake original $\hat{I}'$ with a confidence value of 23.02.

Figure 4 shows the true original, the watermarked image, and a fake original. There are no visible artifacts observed from looking at these three images. They are virtually identical.

□

In the previous example, not only is the presence of our fabricated watermark well above random, it is virtually the same as the presence of the real watermark in the fabricated image! Based on the test results, who is the rightful owner of this image? Which image is the true original, and which is the fake original? Under such circumstances, what can invisible watermarks achieve? There is no additional evidence available to support any answers to these questions, and consequently, invalidating the claims of rightful ownerships.

The method of attack described above is universal in the sense that any image watermarked by *any* scheme can be defeated. In the absence of *standardization* on the invisible watermarking techniques, or any specification of requirements on legitimate watermarking schemes, any one can claim ownership of any watermarked image he or she has access to.

[3]a simple modification—for a 500-line C program, a single '*' was changed to a '/'.

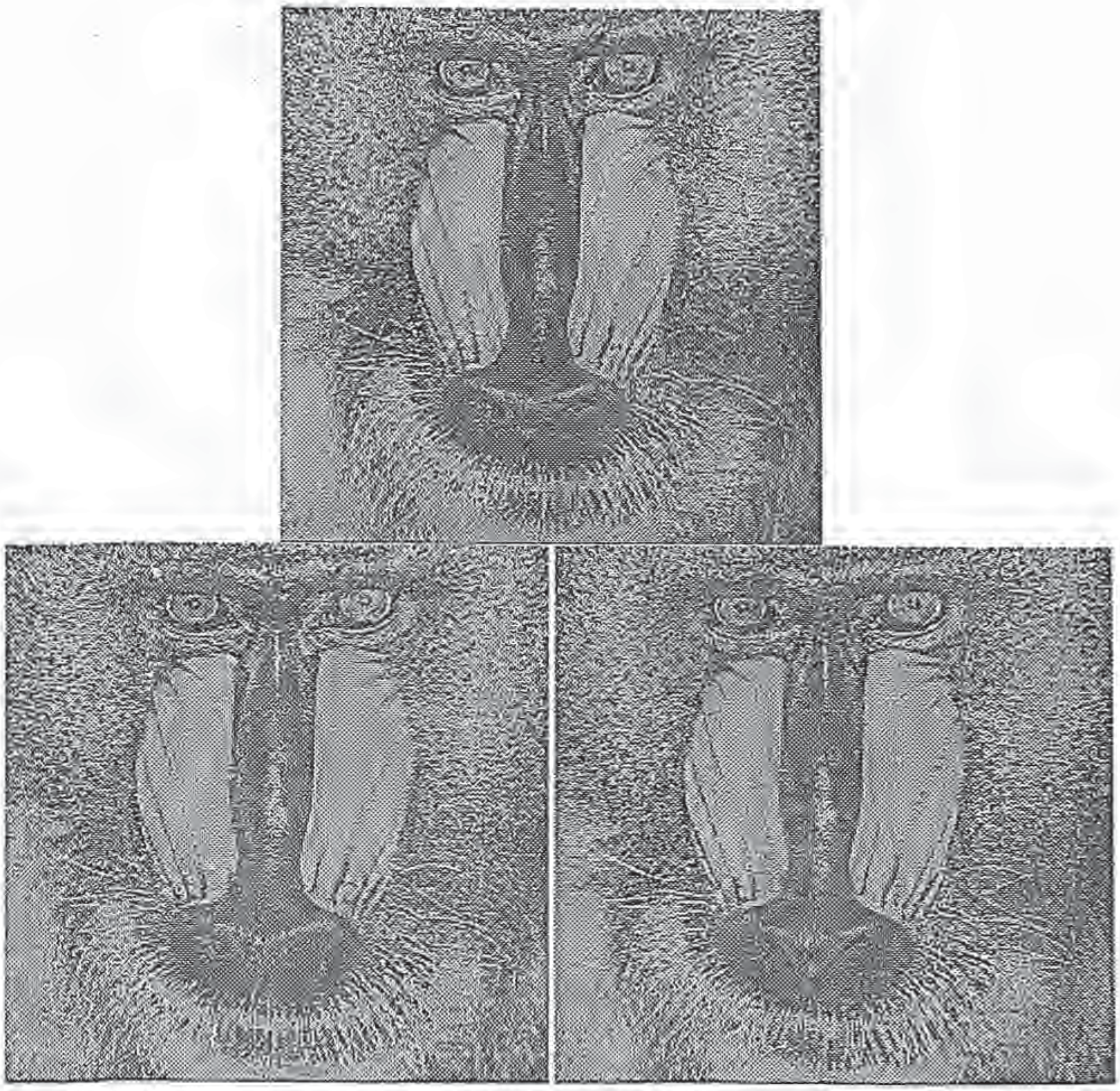


Figure 3: Three "Baboon" images (from USC database). (TOP CENTER) the watermarked image ($\hat{I}$) of the original with 1000-element watermark sequence inserted. (BOTTOM LEFT) The original image I. (BOTTOM RIGHT) The fabricated "original" image $\hat{I}'$. Confidence measure of the original watermark S in image $\hat{I}'$ is 23.02. Confidence measure of the fabricated watermark S' in image I is 23.52.

No matter which scheme Alice uses to invisibly watermark her image, Bob can always use an invertible watermarking scheme $(\mathcal{E}, \mathcal{D}, C_\delta)$ (such as the one in Example 3) to create a counterfeit original (that is, he uses $\mathcal{E}'$ to create a fake original $\hat{I}'$, and can then show that this image, when watermarked with S' using $\mathcal{E}$, will give the watermarked image $\hat{I}$ as in circulation) and proceed to argue that the unique ownership cannot be determined — thus Alice's claim of ownership is not validated based solely on the test of the presence of her invisible watermarks.

5 Non-invertible Watermarking of Images

In the previous section we have shown how one can fabricate an "original" image from a watermarked image such that rightful ownership cannot be resolved. We call the class of watermarking scheme that can be attacked by creating a "counterfeit original" as the *invertible* watermarking schemes. A more precise definition of the class of invertible watermarking schemes is as follows:

Definition: Invertible Watermarking Schemes

Let $\mathcal{V}$ be the class of invertible (invisible) watermarking schemes. Let S be a digital watermark, and S' be another digital watermark, such that both S and S' belongs to the set of allowable watermarks, then given an image I, $(\mathcal{E}, \mathcal{D}, C_\delta) \in \mathcal{V}$ if for any S and S', there exists a $(\mathcal{E}', \mathcal{D}, C_\delta)$ with

$$\mathcal{E}'(\hat{I}, S') = \hat{I}'$$

and

$$\mathcal{E}(\hat{I}', S') = \hat{I},$$

where $\hat{I} = \mathcal{E}(I, S)$, such that $\mathcal{E}'$ is a *computationally feasible* encoding function, and there is no perceptual quality degradation in the derivative images $\hat{I}$ and $\hat{I}'$ from I.

Otherwise, $(\mathcal{E}, \mathcal{D}, C_\delta)$ is *non-invertible*. □

Note that this definition does not put any requirements on the decoding function $\mathcal{D}$.

While what we have described so far is a wide-sweeping attack, applicable to most current digital watermarking schemes, it seems that it may be foiled through more careful requirements for, or standardization of, the watermarking schemes. In other words, we can require that from legal point of view, to establish rightful ownership through invisible watermarks, the watermarking schemes applied to the images have to satisfy certain requirements — among them, the watermarking schemes cannot be invertible.

There are a number of ways to enforce this new requirement. One could develop a method which can be inverted, but in such a way that image quality is degraded to a high enough degree that the fabricated "original" image is clearly not the real original. In fact, it has been suggested that in the above process of signature fabrication, the image may already be degraded just enough to make it clear to an expert what is the real original and what is the fake one. However, progress in invisible watermarking schemes will most likely yield methods that create even less of a degradation on image quality, to the point that a fake original image as computed above (really, the result of watermarking an image *twice*, using two slightly different schemes), will be visually indistinguishable from the original. In fact, as Figure 4 shows, this is already true — there are no visible artifacts in the fake originals we fabricated. Finally, there are some images (pictures of clouds, say, or abstract art) for which comparisons of quality between original and modified versions will be difficult. In short, judgements based on perceptual "quality" are weak and unreliable for resolving image ownership.

Another approach would be to use one-way functions in the watermarking process, i.e., it would not possible to extract the watermark once it is inserted. The presence will then have to be inferred. Such a method, hopefully, does not suffer considerably in terms of flexibility or efficiency. With reference to the general formulation for watermarking described in Section 2, making the insertion operation non-invertible is an undesirable move, even though that insertion operation's invertibility is the source of our problems. This is because the confidence measure relies on the fact that the watermark vector can be later extracted, and using a non-invertible insertion formula may make this difficult. If the transform $D()$ that maps the image into feature space is linear, then it may be possible to extract a signature without using the extraction operator $\ominus$. However, in general this may not be always true.

A third approach follows from noting that in order to fabricate a counterfeit original $\hat{I}'$, the attacker (Bob) first chooses a watermark S' that he proceeds to "remove" from the watermarked image $\hat{I}$ that belongs to the owner (Alice). Now, if we enforce the requirement that any watermark S that is inserted into an image I be dependent on I, then we make it difficult for Bob to select S' as it depends on $\hat{I}'$ which has not yet been determined. This can be achieved by computing a bit sequence $B = \{b_1, b_2, \ldots\}$ from the image I that is being watermarked and using these bits in the watermarking process. One way of using the bits is to select the labels s_i that compose the watermark itself. Another way is to use the bits to choose between two different insertion operations $\oplus$ and $\otimes$. Furthermore, obtaining the bit sequence from a one-way function of the image to be watermarked, would make the scheme secure against any trial and error type of attacks that Bob can attempt in order to construct a counterfeit original. We now provide one such example and discuss possible attacks using the *same* scheme.

Example 4 *A modified version of the scheme described by Cox et. al. [1].*

We first produce a 1000-bit $\{b_1, b_2, \cdots b_{1000}\}$ one-way hash of the original image before computing the 2D DCT of the image. We then use two slightly different equations for inserting the watermark vector elements. For each frequency bin v_i to be modified, we choose one of the two formulas depending on the value of the hash bit b_i. The formulas are chosen to be different enough in their output that a watermark vector consisting of the same vector elements, but using a different 1000-bit hash string, will not show up in the watermarked image.

Specifically, we used two versions of the second update formula in [1] as follows:

$$v_i' = \begin{cases} v_i(1+\alpha s_i), & b_i = 0; \\ v_i(1-\alpha s_i), & b_i = 1, \end{cases} \tag{13}$$

where in both cases α was chosen to be 0.1. A 1000-bit hash of the image was computed, and for each of the 1000 highest AC DCT matrix elements, one of the formula was used depending on the value of the hash bit b_i. For convenience, these hash bits were stored in the watermark vector file.

Anticipating a possible attack involving rearranging watermark elements to match the required hash values, our scheme requires that the elements be embedded in the high-magnitude matrix elements in a left-to-right, top-to-bottom order. In addition, we can impose the requirement that $s_i > 0$ for otherwise, an attacker can simply negate certain watermark vector elements to match the resulting hash bits, i.e., an attacker chooses an arbitrary binary sequence $a_1, a_2, \cdots, a_{1000}$ and applies the following transformation on the watermarked image $\hat{I}$ to create an "original" $\hat{I}'$:

$$v_i'' = \begin{cases} v_i'(1 - \alpha \hat{s}_i), & a_i = 0; \\ v_i'(1 + \alpha \hat{s}_i), & a_i = 1. \end{cases} \tag{14}$$

He then computes the hash of his "original" $\hat{I}'$: $\{\hat{b}_1', \hat{b}_2', \cdots\}$. His watermark is then given as follows:

$$\hat{s}_i' = \begin{cases} \hat{s}_i, & a_i = \hat{b}_i'; \\ -\hat{s}_i, & \text{otherwise.} \end{cases} \tag{15}$$

This could also be avoided by choosing two update formulas that differ in a different way than the two we used. It is also important that the hash string be image-dependent for this scheme to be robust against attacks.

We apply this watermarking scheme on a test image. The original watermark S is applied 1000 times, once with an original 1000-bit hash string, and the other 999 times using randomly selected bit strings. The 1000 different watermarked images are tested for the presence of the signature S. The results are displayed in Figure 4. As illustrated in the figure, if the 1000-bit hash of the "original" hash string cannot be anticipated, the resulting watermark cannot be expected to have a high presence and is thus useless. □

The above example illustrates the difficulty in "inverting" the watermarking scheme. Because the fake "original", $\hat{I}'$, has not been recreated, one cannot know the associated hash string $\{\hat{b}_1', \hat{b}_2', \cdots\}$. This in turn implies that it is not possible to decide which formula to use to "remove" a watermark element from $\hat{I}$. We believe the proposed scheme is non–invertible although it seems difficult to rigorously prove this.

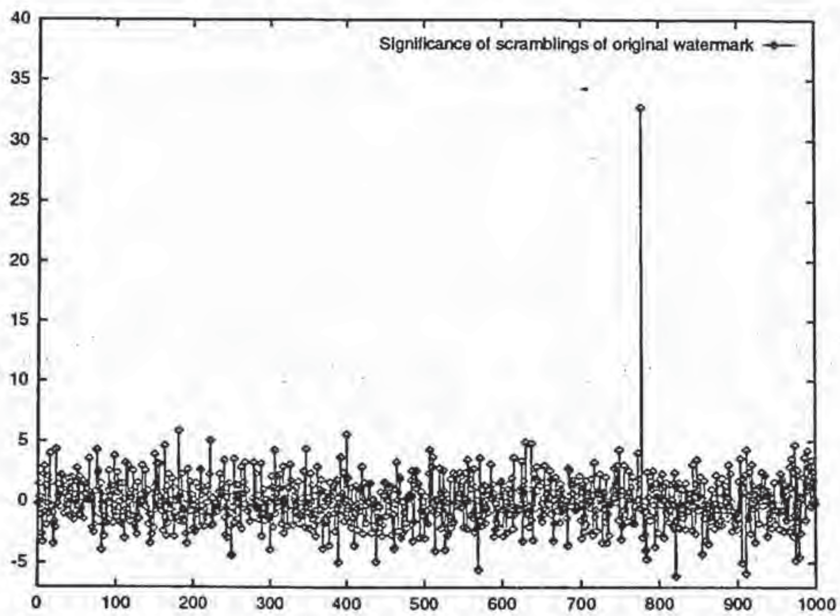


Figure 4: Results of scrambling hash bits in watermark vectors. The original (as seen by the spike) and 999 copies with scrambled hashes

In conclusion, requiring non-invertibility of a watermarking scheme may be necessary to prevent fabrication attacks such as those we just illustrated. Without such requirements, one could use a non-invertible scheme such as the one described in Section 4 to generate the necessary confusion. Thus, from a legal standpoint, to be able to establish rightful ownership through invisible watermarks, it would be necessary that watermarking schemes satisfy certain requirements. We have demonstrated in this section that non-invertibility could be an answer.

6 Conclusions and Discussions

We now return to the questions that we posed in the beginning. What is a watermark, why is it needed and how useful is it? Current copyrighting mechanisms for photographs and images involve registration of the item being copyrighted with a centralized authority[4]. All contests of ownership are then resolved by this central authority. It has been recognized for quite some time now that these laws are quite inadequate for dealing with digital data which can be so easily copied and manipulated. This led to an interest within the research community for developing copyright protection mechanisms. One such effort was aimed at developing watermarking techniques for digital data. A watermark in this context is a signal added to digital data such that it could be used to (1) identify source of the data or uniquely establish ownership, (2) to identify its intended recipient, and (3) to check if the data has been tampered with. Within each class of applications, there could be variations on the requirement of the watermarking scheme.

It may have appeared from some of the ensuing work that the most important property of such watermarking schemes was their robustness. That is their ability to survive despite malicious attempts at removal. Indeed, in this sense the research efforts have been very successful. Watermarking schemes have been proposed and shown to be remarkably robust. However, we have demonstrated in this paper that the ability to embed robust watermarks in digital images does not necessarily imply the ability to establish ownership, unless certain

[4]We quote several sentences in [7] here: in U.S. copyright laws, copyright protection is secured automatically when the work is "created", and a work is "created" when it is fixed in a copy or phonorecord for the first time. No publication or registration or other action in the Copyright Office is required to secure the copyrights. There are, however, certain definite advantages to registration.

requirements are imposed legally on the watermarking schemes. In the absence of such requirements, Alice cannot simply lock away an original image which she can use later to establish ownership over a watermarked copy. So what can Alice do? She can still resort to conventional means of registering the image with a central authority and obtaining a copyright.

But then, what would be the need of watermarking? Watermarking would still be very much needed for protecting her interests. For example, Alice could embed a different watermark in each copy of the image that she sells. This unique watermark would enable her to determine the identity of her specific customer who may be making unauthorized copies and selling them for a profit. A watermark could also be used by Alice to establish her ownership over versions of her image that have been visually modified. For example, Alice can establish that it is her image that is embedded in a larger image. Current copyrighting mechanisms are not well geared for addressing such situations, which in the future can easily arise, given the ease by which data in digital form can be manipulated and the widespread use of the Internet in rapid dissemination of digital information.

One can list many more variants of such applications demonstrating the utility of invisible watermarks. However, different applications would require different types of watermarking schemes with different requirements. For example, non-invertibility may not be an issue when there is a centralized authority with whom copyrighted images are registered. However, in such an application it would be useful for an user to query an image for copyright protection. In certain applications such as the use of invisible watermark to ensure the integrity of digital fingerprints, it is important only to verify that a fingerprint image has not been tampered with; in such a case, the robustness of the watermark against image modification is *not* an issue. On the other hand, it is also important to investigate into crytographic protocols that make it difficult for general attacks (such as that proposed in [5]) to *remove* or *diminish the presence of* the watermark in the image.

References

[1] I.J. Cox, J. Kilian, T. Leighton and T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia", *NEC Research Institute Technical Report*, 95-10.

[2] J. Zhao and E. Koch, "Embedding Robust Labels into images for Copyright Protection", *Intellectual Property Rights and New Technologies, Proceedings of the KnowRight'95 Conference 1995*, pp.242–51 .

[3] N. Nikolaidis and I. Pitas, "Copyright Protection of Images using Robust Digital Signatures", *Proceedings, IEEE International Conference on Acoustics, Speech and Signal Processing 1996.*

[4] F.M. Boland, J.J.K. O'Ruandaidh and C. Dautzenberg, "Watermarking Digital Images for Copyright Protection", *Proceedings, IEE Image Processing And its Applications Conference 1995*, pp.326–330.

[5] H.S. Stone, "Analysis of Attacks on Image Watermarks with Randomized Coefficients", *NEC Research Institute Technical Report*, 1996.

[6] G. Braudaway, K. Magerlein and F. Mintzer, "Protecting publicly available images with a visible image watermark", *Proc. SPIE: Optical Security and Counterfeit Deterrence Techniques*, Vol. 2659, pp. 126–133, 1996.

[7] U.S. Copyright Office, Circular 1, "Copyright Basics", March 28, 1996. The circular is available on Internet via World Wide Web URL *http://lcweb.loc.gov/copyright* and via Gopher *marvel.loc.gov*.

Numerical Recipes In C 398

Chapter 12. Fourier Transform Spectral Methods

12.0 Introduction

A very large class of important computational problems falls under the general rubric of "Fourier transform methods" or "spectral methods." For some of these problems, the Fourier transform is simply an efficient computational tool for accomplishing certain common manipulations of data. In other cases, we have problems for which the Fourier transform (or the related "power spectrum") is itself of intrinsic interest. These two kinds of problems share a common methodology.

Largely for historical reasons the literature on Fourier and spectral methods has been disjoint from the literature on "classical" numerical analysis. In this day and age there is no justification for such a split. Fourier methods are commonplace in research and we shall not treat them as specialized or arcane. At the same time, we realize that many computer users have had relatively less experience with this field than with, say, differential equations or numerical integration. Therefore our summary of analytical results will be more complete. Numerical algorithms, per se, begin in §12.2.

A physical process can be described either in the *time domain*, by the values of some quantity h as a function of time t, e.g. $h(t)$, or else in the *frequency domain*, where the process is specified by giving its amplitude H (generally a complex number indicating phase also) as a function of frequency f, that is $H(f)$, with $-\infty < f < \infty$. For many purposes it is useful to think of $h(t)$ and $H(f)$ as being two different *representations* of the *same* function. One goes back and forth between these two representations by means of the *Fourier transform* equations,

$$H(f) = \int_{-\infty}^{\infty} h(t)e^{2\pi i f t}dt \qquad h(t) = \int_{-\infty}^{\infty} H(f)e^{-2\pi i f t}df \tag{12.0.1}$$

If t is measured in seconds, then f in equation (12.0.1) is in cycles per second, or Hertz (the unit of frequency). However, the equations work with other units. If h is a function of position x (in meters), H will be a function of inverse wavelength (cycles per meter), and so on. If you are trained as a physicist or mathematician, you are probably more used to using *angular frequency* ω, which is given in *radians* per sec. The relation between ω and f, $H(\omega)$ and $H(f)$ is

$$\omega \equiv 2\pi f \qquad H(\omega) \equiv [H(f)]_{f=\omega/2\pi} \tag{12.0.2}$$

and equation (12.0.1) looks like this

$$H(\omega) = \int_{-\infty}^{\infty} h(t)e^{i\omega t}dt \qquad h(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} H(\omega)e^{-i\omega t}d\omega \tag{12.0.3}$$

We were raised on the ω-convention, but we changed! There are fewer factors of 2π to remember if you use the f-convention, especially when we get to discretely sampled data in §12.1.

From equation (12.0.1) it is evident at once that Fourier transformation is a *linear* operation. The transform of the sum of two functions is equal to the sum of the transforms. The transform of a constant times a function is that same constant times the transform of the function.

In the time domain, function $h(t)$ may happen to have one or more special symmetries It might be *purely real* or *purely imaginary* or it might be *even*, $h(t) = h(-t)$, or *odd*, $h(t) = -h(-t)$. In the frequency domain, these symmetries lead to relationships between $H(f)$ and $H(-f)$. The following table gives the correspondence between symmetries in the two domains:

If...	then...
$h(t)$ is real	$H(-f) = [H(f)]^*$
$h(t)$ is imaginary	$H(-f) = -[H(f)]^*$
$h(t)$ is even	$H(-f) = H(f)$ [i.e. $H(f)$ is even]
$h(t)$ is odd	$H(-f) = -H(f)$ [i.e. $H(f)$ is odd]
$h(t)$ is real and even	$H(f)$ is real and even
$h(t)$ is real and odd	$H(f)$ is imaginary and odd
$h(t)$ is imaginary and even	$H(f)$ is imaginary and even
$h(t)$ is imaginary and odd	$H(f)$ is real and odd

In subsequent sections we shall see how to use these symmetries to increase computational efficiency.

Here are some other elementary properties of the Fourier transform. (We'll use the "$\Longleftrightarrow$" symbol to indicate transform pairs.) If

$$h(t) \Longleftrightarrow H(f)$$

0280 0281

is such a pair, then other transform pairs are

$$h(at) \Longleftrightarrow \frac{1}{|a|}H\left(\frac{f}{a}\right) \qquad \text{"time scaling"} \qquad (12.0.4)$$

$$\frac{1}{|b|}h\left(\frac{t}{b}\right) \Longleftrightarrow H(bf) \qquad \text{"frequency scaling"} \qquad (12.0.5)$$

$$h(t-t_0) \Longleftrightarrow H(f)\,e^{2\pi i f t_0} \qquad \text{"time shifting"} \qquad (12.0.6)$$

$$h(t)\,e^{-2\pi i f_0 t} \Longleftrightarrow H(f-f_0) \qquad \text{"frequency shifting"} \qquad (12.0.7)$$

With two functions $h(t)$ and $g(t)$, and their corresponding Fourier transforms $H(f)$ and $G(f)$, we can form two combinations of special interest. The *convolution* of the two functions, denoted $g * h$, is defined by

$$g * h \equiv \int_{-\infty}^{\infty} g(\tau)h(t-\tau)\,d\tau \qquad (12.0.8)$$

Note that $g * h$ is a function in the time domain and that $g * h = h * g$. It turns out that the function $g * h$ is one member of a simple transform pair

$$g * h \Longleftrightarrow G(f)H(f) \qquad \text{"Convolution Theorem"} \qquad (12.0.9)$$

In other words, the Fourier transform of the convolution is just the product of the individual Fourier transforms.

The *correlation* of two functions, denoted $\mathrm{Corr}(g,h)$, is defined by

$$\mathrm{Corr}(g,h) \equiv \int_{-\infty}^{\infty} g(\tau+t)h(\tau)\,d\tau \qquad (12.0.10)$$

The correlation is a function of t, which is called the *lag*. It therefore lies in the time domain, and it turns out to be one member of the transform pair:

$$\mathrm{Corr}(g,h) \Longleftrightarrow G(f)H^*(f) \qquad \text{"Correlation Theorem"} \qquad (12.0.11)$$

[More generally, the second member of the pair is $G(f)H(-f)$, but we are restricting ourselves to the usual case in which g and h are real functions, so we take the liberty of setting $H(-f) = H^*(f)$.] This result shows that multiplying the Fourier transform of one function by the complex conjugate of the Fourier Transform of the other gives the Fourier transform of their correlation. The correlation of a function with itself is called its *autocorrelation*. In this case (12.0.11) becomes the transform pair

$$\mathrm{Corr}(g,g) \Longleftrightarrow |G(f)|^2 \qquad \text{"Wiener-Khinchin Theorem"} \qquad (12.0.12)$$

The *total power* in a signal is the same whether we compute it in the time domain or in the frequency domain. This result is known as *Parseval's theorem*:

$$\text{Total Power} \equiv \int_{-\infty}^{\infty} |h(t)|^2\,dt = \int_{-\infty}^{\infty} |H(f)|^2\,df \qquad (12.0.13)$$

Frequently one wants to know "how much power" is contained in the frequency interval between f and $f+df$. In such circumstances one does not usually distinguish between positive and negative f, but rather regards f as varying from 0 ("zero frequency" or D.C.) to $+\infty$. In such cases, one defines the *one-sided power spectral density (PSD)* of the function h as

$$P_h(f) \equiv |H(f)|^2 + |H(-f)|^2 \qquad 0 \le f < \infty \qquad (12.0.14)$$

so that the total power is just the integral of $P_h(f)$ from $f=0$ to $f=\infty$. When the function $h(t)$ is real, then the two terms in (12.0.14) are equal, so $P_h(f) = 2|H(f)|^2$. Be warned that one occasionally sees PSDs defined without this factor two. These, strictly speaking, are called *two-sided power spectral densities*, but some books are not careful about stating whether one- or two-sided is to be assumed. We will always use the one-sided density given by equation (12.0.14). Figure 12.0.1 contrasts the two conventions.

If the function $h(t)$ goes endlessly from $-\infty < t < \infty$, then its total power and power spectral density will, in general, be infinite. Of interest then is the *(one- or two-sided) power spectral density per unit time*. This is computed by taking a long, but finite, stretch of the function $h(t)$, computing its PSD [that is, the PSD of a function which equals $h(t)$ in the finite stretch but is zero everywhere else], and then dividing the resulting PSD by the length of the stretch used. Parseval's theorem in this case states that the integral of the one-sided PSD-per-unit-time over positive frequency is equal to the mean-square amplitude of the signal $h(t)$.

You might well worry about how the PSD-per-unit-time, which is a function of frequency f, converges as one evaluates it using longer and longer stretches of data. This interesting question is the content of the subject of "power spectrum estimation," and will be considered below in §12.8–§12.9. A crude answer for now is: the PSD-per-unit-time converges to finite values at all frequencies *except* those where $h(t)$ has a discrete sine-wave (or cosine-wave) component of finite amplitude. At those frequencies, it becomes a delta-function, i.e. a sharp spike, whose width gets narrower and narrower, but whose area converges to be the mean-square amplitude of the discrete sine or cosine component at that frequency.

We have by now stated all of the analytical formalism that we will need in this chapter with one exception: In computational work, especially with experimental data, we are almost never given a continuous function $h(t)$ to work with, but are given, rather, a list of measurements of $h(t_i)$ for a discrete

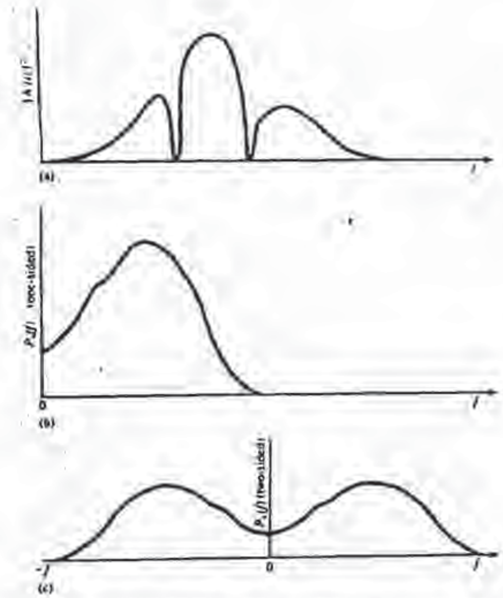


Figure 12.0.1 Normalizations of one- and two-sided power spectra. The area under the square of the function, (a), equals the area under its one-sided power spectrum at positive frequencies, (b), and also equals the area under its two-sided power spectrum at positive and negative frequencies, (c).

set of t_i's. The profound implications of this seemingly unimportant fact are the subject of the next section.

REFERENCES AND FURTHER READING:

Champeney, D.C. 1973, *Fourier Transforms and Their Physical Applications* (New York: Academic Press).

Elliott, D.F., and Rao, K.R. 1982, *Fast Transforms: Algorithms, Analyses, Applications* (New York: Academic Press).

12.1 Fourier Transform of Discretely Sampled Data

In the most common situations, function $h(t)$ is sampled (i.e., its value is recorded) at evenly spaced intervals in time. Let Δ denote the time interval between consecutive samples, so that the sequence of sampled values is

$$h_n = h(n\Delta) \qquad n = \ldots, -3, -2, -1, 0, 1, 2, 3, \ldots \tag{12.1.1}$$

The reciprocal of the time interval Δ is called the *sampling rate*; if Δ is measured in seconds, for example, then the sampling rate is the number of samples recorded per second.

Sampling Theorem and Aliasing

For any sampling interval Δ, there is also a special frequency f_c, called the *Nyquist critical frequency*, given by

$$f_c \equiv \frac{1}{2\Delta} \tag{12.1.2}$$

If a sine wave of the Nyquist critical frequency is sampled at its positive peak value, then the next sample will be at its negative trough value, the sample after that at the positive peak again, and so on. Expressed otherwise: *Critical sampling of a sine wave is two sample points per cycle.* One frequently chooses to measure time in units of the sampling interval Δ. In this case the Nyquist critical frequency is just the constant 1/2.

The Nyquist critical frequency is important for two related, but distinct, reasons. One is good news, and the other bad news. First the good news. It is the remarkable fact known as the *sampling theorem*: If a continuous function $h(t)$, sampled at an interval Δ, happens to be *band-width limited* to frequencies smaller in magnitude than f_c, i.e., if $H(f) = 0$ for all $|f| > f_c$, then the function $h(t)$ is *completely determined* by its samples h_n. In fact, $h(t)$ is given explicitly by the formula

$$h(t) = \Delta \sum_{n=-\infty}^{+\infty} h_n \frac{\sin[2\pi f_c(t - n\Delta)]}{\pi(t - n\Delta)} \tag{12.1.3}$$

This is a remarkable theorem for many reasons, among them that it shows that the "information content" of a band-width limited function is, in some sense, infinitely smaller than that of a general continuous function. Fairly often, one is dealing with a signal which is known on physical grounds to be band-width limited (or at least approximately band-width limited). For example, the signal may have passed through an amplifier with a known, finite frequency response. In this case, the sampling theorem tells us that the entire information content of the signal can be recorded by sampling it at a rate Δ^{-1} equal to twice the maximum frequency passed by the amplifier (cf. 12.1.2).

Now the bad news. The bad news concerns the effect of sampling a continuous function that is *not* band-width limited to less than the Nyquist critical frequency. In that case, it turns out that all of the power spectral density which lies outside of the frequency range $-f_c < f < f_c$ is spuriously moved into that range. This phenomenon is called *aliasing*. Any frequency component outside of the frequency range $(-f_c, f_c)$ is *aliased* (falsely translated) into that range by the very act of discrete sampling. You can readily convince yourself that two waves $\exp(2\pi i f_1 t)$ and $\exp(2\pi i f_2 t)$ give the same samples at an interval Δ if and only if f_1 and f_2 differ by a multiple of $1/\Delta$, which is just the width in frequency of the range $(-f_c, f_c)$. There is little that you can do to remove aliased power once you have discretely sampled a signal. The way to overcome aliasing is to (i) know the natural band-width limit of the signal—or else enforce a known limit by analog filtering of the continuous signal, and then (ii) sample at a rate sufficiently rapid to give two points per cycle of the highest frequency present. Figure 12.1.1 illustrates these considerations.

To put the best face on this, we can take the alternative point of view: If a continuous function has been competently sampled, then, when we come to estimate its Fourier transform from the discrete samples, we can *assume* (or rather we *might as well* assume) that its Fourier transform is equal to zero outside of the frequency range in between $-f_c$ and f_c. Then we look to the Fourier transform to tell whether the continuous function *has* been competently sampled (aliasing effects minimized). We do this by looking to see whether the Fourier transform is already approaching zero as the frequency approaches f_c from below, or $-f_c$ from above. If, on the contrary, the transform is going towards some finite value, then chances are that components outside of the range have been folded back over onto the critical range.

Discrete Fourier Transform

We now estimate the Fourier transform of a function from a finite number of its sampled points. Suppose that we have N consecutive sampled values

$$h_k \equiv h(t_k), \qquad t_k \equiv k\Delta, \qquad k = 0, 1, 2, \ldots N-1 \tag{12.1.4}$$

so that the sampling interval is Δ. To make things simpler, let us also suppose that N is even. If the function $h(t)$ is nonzero only in a finite interval of time, then that whole interval of time is supposed to be contained in the range of the N points given. Alternatively, if the function $h(t)$ goes on forever, then the sampled points are supposed to be at least "typical" of what $h(t)$ looks like at all other times.

With N numbers of input, we will evidently be able to produce no more than N independent numbers of output. So, instead of trying to estimate the

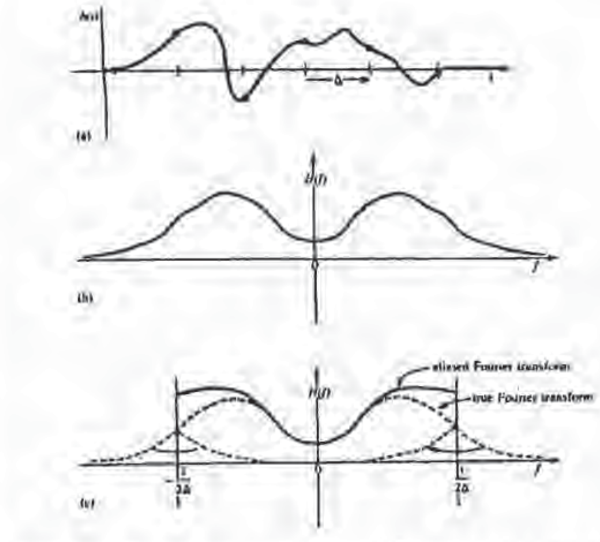


Figure 12.1.1. The continuous function shown in (a) is nonzero only for a finite interval of time T. It follows that its Fourier transform, shown schematically in (b), is not bandwidth limited but has finite amplitude for all frequencies. If the original function is sampled with a sampling interval Δ, as in (a), then the Fourier transform (c) is defined only between plus and minus the Nyquist critical frequency. Power outside that range is folded over or "aliased" into the range. The effect can be eliminated only by low-pass filtering the original function *before sampling*.

Fourier transform $H(f)$ at all values of f in the range $-f_c$ to f_c, let us seek estimates only at the discrete values

$$f_n \equiv \frac{n}{N\Delta}, \qquad n = -\frac{N}{2}, \ldots, \frac{N}{2} \tag{12.1.5}$$

The extreme values of n in (12.1.5) correspond exactly to the lower and upper limits of the Nyquist critical frequency range. If you are really on the ball, you will have noticed that there are $N+1$, not N, values of n in (12.1.5); it will turn out that the two extreme values of n are not independent (in fact they are equal), but all the others are. This reduces the count to N.

The remaining step is to approximate the integral in (12.0.1) by a discrete sum:

$$H(f_n) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f_n t} dt \approx \sum_{k=0}^{N-1} h_k e^{2\pi i f_n t_k} \Delta = \Delta \sum_{k=0}^{N-1} h_k e^{2\pi i k n/N} \tag{12.1.6}$$

Here equations (12.1.4) and (12.1.5) have been used in the final equality. The final summation in equation (12.1.6) is called the *discrete Fourier transform* of the N points h_k. Let us denote it by H_n,

$$H_n \equiv \sum_{k=0}^{N-1} h_k e^{2\pi i k n/N} \tag{12.1.7}$$

The discrete Fourier transform maps N complex numbers (the h_k's) into N complex numbers (the H_n's). It does not depend on any dimensional parameter, such as the time scale Δ. The relation (12.1.6) between the discrete Fourier transform of a set of numbers and their continuous Fourier transform when they are viewed as samples of a continuous function sampled at an interval Δ can be rewritten as

$$H(f_n) \approx \Delta H_n \tag{12.1.8}$$

where f_n is given by (12.1.5).

Up to now we have taken the view that the index n in (12.1.7) varies from $-N/2$ to $N/2$ (cf. 12.1.5). You can easily see, however, that (12.1.7) is periodic in n, with period N. Therefore, $H_{-n} = H_{N-n}$ $n = 1, 2, \ldots$. With this conversion in mind, one generally lets the n in H_n vary from 0 to $N-1$ (one complete period). Then n and k (in h_k) vary exactly over the same range, so the mapping of N numbers into N numbers is manifest. When this convention is followed, you must remember that zero frequency corresponds to $n = 0$, positive frequencies $0 < f < f_c$ correspond to values $1 \le n \le N/2 - 1$, while negative frequencies $-f_c < f < 0$ correspond to $N/2 + 1 \le n \le N - 1$. The value $n = N/2$ corresponds to *both* $f = f_c$ and $f = -f_c$.

The discrete Fourier transform has symmetry properties almost exactly the same as the continuous Fourier transform. For example, all the symmetries in the table following equation (12.0.3) hold if we read h_k for $h(t)$, H_n for $H(f)$, and H_{N-n} for $H(-f)$. (Likewise, "even" and "odd" in time refer to whether the values h_k at k and $N-k$ are identical or the negative of each other.)

The formula for the discrete *inverse* Fourier transform, which recovers the set of h_k's exactly from the H_n's is:

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\pi i k n/N} \tag{12.1.9}$$

Notice that the only differences between (12.1.9) and (12.1.7) are (i) changing the sign in the exponential, and (ii) dividing the answer by N. This means that a routine for calculating discrete Fourier transforms can also, with slight modification, calculate the inverse transforms.

The discrete form of Parseval's theorem is

$$\sum_{k=0}^{N-1} |h_k|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |H_n|^2 \tag{12.1.10}$$

There are also discrete analogs to the convolution and correlation theorems (equations 12.0.9 and 12.0.11), but we shall defer them to §12.4 and §12.5, respectively.

REFERENCES AND FURTHER READING:

Brigham, E. Oran. 1974, *The Fast Fourier Transform* (Englewood Cliffs, N.J.: Prentice-Hall).

Elliott, D.F., and Rao, K.R. 1982, *Fast Transforms: Algorithms, Analyses, Applications* (New York: Academic Press).

12.2 Fast Fourier Transform (FFT)

How much computation is involved in computing the discrete Fourier transform (12.1.7) of N points? For many years, until the mid-1960s, the standard answer was this: Define W as the complex number

$$W \equiv e^{2\pi i/N} \tag{12.2.1}$$

Then (12.1.7) can be written as

$$H_n = \sum_{k=0}^{N-1} W^{nk} h_k \tag{12.2.2}$$

In other words, the vector of h_k's is multiplied by a matrix whose $(n,k)^{th}$ element is the constant W to the power $n \times k$. The matrix multiplication produces a vector result whose components are the H_n's. This matrix multiplication evidently requires N^2 complex multiplications, plus a smaller number of operations to generate the required powers of W. So, the discrete Fourier transform appears to be an $O(N^2)$ process. These appearances are deceiving! The discrete Fourier transform can, in fact, be computed in $O(N \log_2 N)$ operations with an algorithm called the *Fast Fourier Transform*, or *FFT*. The difference between $N \log_2 N$ and N^2 is immense. With $N = 10^6$, for example, it is the difference between, roughly, 30 seconds of CPU time and 2 weeks of CPU time on a microsecond cycle time computer. The existence of an FFT algorithm became generally known only in the mid-1960s, from the work of J.W. Cooley and J.W. Tukey, who in turn had been prodded by R.L. Garwin of IBM Yorktown Heights Research Center. Retrospectively, we now know that a few clever individuals had independently discovered, and in some cases implemented, fast Fourier transforms as many as 20 years previously (see Brigham for references).

One of the earliest "discoveries" of the FFT, that of Danielson and Lanczos in 1942, still provides one of the clearest derivations of the algorithm. Danielson and Lanczos showed that a discrete Fourier transform of length N can be rewritten as the sum of two discrete Fourier transforms, each of length $N/2$. One of the two is formed from the even-numbered points of the original N, the other from the odd-numbered points. The proof is simply this:

$$\begin{aligned} F_k &= \sum_{j=0}^{N-1} e^{2\pi i jk/N} f_j \\ &= \sum_{j=0}^{N/2-1} e^{2\pi i k(2j)/N} f_{2j} + \sum_{j=0}^{N/2-1} e^{2\pi i k(2j+1)/N} f_{2j+1} \\ &= \sum_{j=0}^{N/2-1} e^{2\pi i kj/(N/2)} f_{2j} + W^k \sum_{j=0}^{N/2-1} e^{2\pi i kj/(N/2)} f_{2j+1} \\ &= F_k^e + W^k F_k^o \end{aligned} \qquad (12.2.3)$$

In the last line, W is the same complex constant as in (12.2.1), F_k^e denotes the k^{th} component of the Fourier transform of length $N/2$ formed from the even components of the original f_j's, while F_k^o is the corresponding transform of length $N/2$ formed from the odd components. Notice also that k in the last line of (12.2.3) varies from 0 to N, not just to $N/2$. Nevertheless, the transforms F_k^e and F_k^o are periodic in k with length $N/2$. So each is repeated through two cycles to obtain F_k.

The wonderful thing about the *Danielson-Lanczos Lemma* is that it can be used recursively. Having reduced the problem of computing F_k to that of computing F_k^e and F_k^o, we can do the same reduction of F_k^e to the problem of computing the transform of its $N/4$ even-numbered input data and $N/4$

odd-numbered data. In other words, we can define F_k^{ee} and F_k^{eo} to be the discrete Fourier transforms of the points which are respectively even-even and even-odd on the successive subdivisions of the data.

Although there are ways of treating other cases, by far the easiest case is the one in which the original N is an integer power of 2. In fact, we categorically recommend that you *only* use FFTs with N a power of two. If the length of your data set is not a power of two, pad it with zeros up to the next power of two. (We will give more sophisticated suggestions in subsequent sections below.) With this restriction on N, it is evident that we can continue applying the Danielson-Lanczos Lemma until we have subdivided the data all the way down to transforms of length 1. What is the Fourier transform of length one? It is just the identity operation that copies its one input number into its one output slot! In other words, for every pattern of e's and o's (numbering $\log_2 N$ in all), there is a one-point transform that is just one of the input numbers f_n

$$F_k^{eoeeoeo\cdots oee} = f_n \quad \text{for some } n \qquad (12.2.4)$$

(Of course this one-point transform actually does not depend on k, since it is periodic in k with period 1.)

The next trick is to figure out which value of n corresponds to which pattern of e's and o's in equation (12.2.4). The answer is: reverse the pattern of e's and o's, then let $e = 0$ and $o = 1$, and you will have, *in binary* the value of n. Do you see why it works? It is because the successive subdivisions of the data into even and odd are tests of successive low-order (least significant) bits of n. This idea of *bit reversal* can be exploited in a very clever way which, along with the Danielson-Lanczos Lemma, makes FFTs practical: Suppose we take the original vector of data f_j and rearrange it into bit-reversed order (see Figure 12.2.1), so that the individual numbers are in the order not of j, but of the number obtained by bit-reversing j. Then the bookkeeping on the recursive application of the Danielson-Lanczos Lemma becomes extraordinarily simple. The points as given are the one-point transforms. We combine adjacent pairs to get two-point transforms, then combine adjacent pairs of pairs to get 4-point transforms, and so on, until the first and second halves of the whole data set are combined into the final transform. Each combination takes of order N operations, and there are evidently $\log_2 N$ combinations, so the whole algorithm is of order $N \log_2 N$ (assuming, as is the case, that the process of sorting into bit-reversed order is no greater in order than $N \log_2 N$).

This, then, is the structure of an FFT algorithm: It has two sections. The first section sorts the data into bit-reversed order. Luckily this takes no additional storage, since it involves only swapping pairs of elements. (If k_1 is the bit reverse of k_2, then k_2 is the bit reverse of k_1.) The second section has an outer loop which is executed $\log_2 N$ times and calculates, in turn, transforms of length $2, 4, 8, \ldots, N$. For each stage of this process, two nested inner loops range over the subtransforms already computed and the elements of each transform, implementing the Danielson-Lanczos Lemma. The operation is made more efficient by restricting external calls for trigonometric sines and

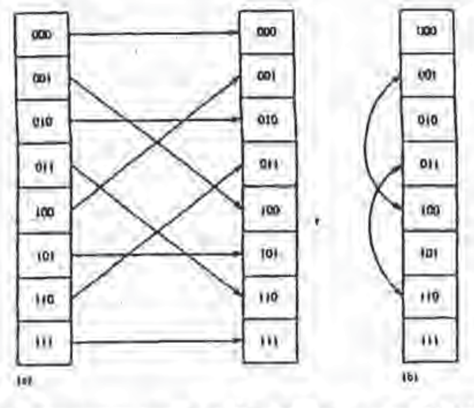


Figure 12.2.1. Reordering an array (here of length 8) by bit reversal, (a) between two arrays, versus (b) in place. Bit reversal reordering is a necessary part of the Fast Fourier Transform (FFT) algorithm.

cosines to the outer loop, where they are made only $\log_2 N$ times. Computation of the sines and cosines of multiple angles is through simple recurrence relations in the inner loops.

The FFT routine given below is based on one originally written by N. Brenner of Lincoln Laboratories. The input quantities are the number of complex data points (nn), the data array (data[1..2*nn]), and isign, which should be set to either ±1 and is the sign of i in the exponential of equation (12.1.7). When isign is set to −1, the routine thus calculates the inverse transform (12.1.9) — except that it does not multiply by the normalizing factor $1/N$ that appears in that equation. You can do that yourself.

Notice that the argument nn is the number of complex data points. The actual length of the real array (data[1..2*nn]) is 2 times nn, with each complex value occupying two consecutive locations. In other words, data[1] is the real part of f_0, data[2] is the imaginary part of f_0, and so on up to data[2*nn-1], which is the real part of f_{N-1}, and data[2*nn] which is the imaginary part of f_{N-1}. The FFT routine gives back the F_n's packed in exactly the same fashion, as nn complex numbers. The real and imaginary parts of the zero frequency component F_0 are in data[1] and data[2]; the smallest nonzero positive frequency has real and imaginary parts in data[3] and data[4]; the smallest (in magnitude) nonzero negative frequency has real and imaginary parts in data[2*nn-1] and data[2*nn]. Positive frequencies increasing in magnitude are stored in the real-imaginary pairs data[5], data[6] up to data[nn-1], data[nn]. Negative frequencies of increasing magnitude are stored in data[2*nn-3], data[2*nn-2] down to data[nn+3], data[nn+4]. Finally, the pair data[nn+1], data[nn+2] contain the real and imaginary parts of the one aliased point which contains the

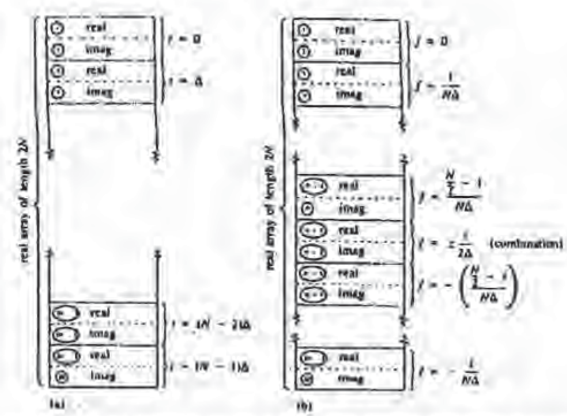


Figure 12.2.2. Input and output arrays for FFT. (a) The input array contains N (a power of 2) complex time samples in a real array of length $2N$, with real and imaginary parts alternating. (b) The output array contains the complex Fourier spectrum at N values of frequency. Real and imaginary parts again alternate. The array starts with zero frequency, works up to the most positive frequency (which is ambiguous with the most negative frequency). Negative frequencies follow, from the second-most negative up to the frequency just below zero.

most positive and the most negative frequency. You should try to develop a familiarity with this storage arrangement of complex spectra, also shown in the Figure 12.2.2, since it is the practical standard.

This is a good place to remind you that you can also use a routine like four1 *without modification* even if your input data array is zero-offset, that is has the range data[0..2*nn-1]. In this case, simply decrement the pointer to data by one when four1 is invoked, e.g. four1(data-1,1024,1);. The real part of f_0 will now be returned in data[0], the imaginary part in data[1], and so on. See §1.2.

```
#include <math.h>

#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr

void four1(data,nn,isign)
float data[];
int nn,isign;
Replaces data by its discrete Fourier transform, if isign is input as 1; or replaces data by
nn times its inverse discrete Fourier transform, if isign is input as -1. data is a complex
array of length nn, input as a real array data[1..2*nn]. nn MUST be an integer power of
2 (this is not checked for!).
{
    int n,mmax,m,j,istep,i;
    double wtemp,wr,wpr,wpi,wi,theta;        Double precision for the trigonometric recurrences.
    float tempr,tempi;
```

DISH-Blue Spike-246
Exhibit 1010, Page 0709

```
n=nn << 1;
j=1;
for (i=1;i<n;i+=2) {                        This is the bit-reversal section of the routine.
    if (j > i) {
        SWAP(data[j],data[i]);              Exchange the two complex numbers.
        SWAP(data[j+1],data[i+1]);
    }
    m=n >> 1;
    while (m >= 2 && j > m) {
        j -= m;
        m >>= 1;
    }
    j += m;
}
mmax=2;                                     Here begins the Danielson-Lanczos section of the routine.
while (n > mmax) {                          Outer loop executed log2 nn times.
    istep=2*mmax;
    theta=6.28318530717959/(isign*mmax);    Initialize for the trigonometric recurrence.
    wtemp=sin(0.5*theta);
    wpr = -2.0*wtemp*wtemp;
    wpi=sin(theta);
    wr=1.0;
    wi=0.0;
    for (m=1;m<mmax;m+=2) {                 Here are the two nested inner loops.
        for (i=m;i<=n;i+=istep) {
            j=i+mmax;                       This is the Danielson-Lanczos formula:
            tempr=wr*data[j]-wi*data[j+1];
            tempi=wr*data[j+1]+wi*data[j];
            data[j]=data[i]-tempr;
            data[j+1]=data[i+1]-tempi;
            data[i] += tempr;
            data[i+1] += tempi;
        }                                   Trigonometric recurrence.
        wr=(wtemp=wr)*wpr-wi*wpi+wr;
        wi=wi*wpr+wtemp*wpi+wi;
    }
    mmax=istep;
}
}
```

Other FFT Algorithms

We should mention that there are a number of variants on the basic FFT algorithm given above. As we have seen, that algorithm first rearranges the input elements into bit-reverse order, then builds up the output transform in $\log_2 N$ iterations. In the literature, this sequence is called a *decimation-in-time* or *Cooley-Tukey* FFT algorithm. It is also possible to derive FFT algorithms which first go through a set of $\log_2 N$ iterations on the input data, and rearrange the *output* values into bit-reverse order. These are called *decimation-in-frequency* or *Sande-Tukey* FFT algorithms. For some applications, such as convolution (§12.4), one takes a data set into the Fourier domain and then, after some manipulation, back out again. In these cases it is possible to avoid all bit reversing. You use a decimation-in-frequency algorithm (without its bit reversing) to get into the "scrambled" Fourier domain, do your operations there, and then use an inverse algorithm (without its bit reversing) to get back to the time domain. While elegant in principle, this procedure does not in practice save much computation time, since the bit reversals represent only a small fraction of an FFT's operations count, and since most useful operations in the frequency domain require a knowledge of which points correspond to which frequencies.

Another class of FFTs subdivides the initial data set of length N not all the way down to the trivial transform of length 1, but rather only down to some other small power of 2, for example $N = 4$, *base-4 FFTs*, or $N = 8$, *base-8 FFTs*. These small transforms are then done by small sections of highly optimized coding which take advantage of special symmetries of that particular small N. For example, for $N = 4$, the trigonometric sines and cosines that enter are all ± 1 or 0, so many multiplications are eliminated, leaving largely additions and subtractions. These can be faster than simpler FFTs by some significant, but not overwhelming, factor, e.g. 20 or 30 percent.

There are also FFT algorithms for data sets of length N not a power of two. They work by using relations analogous to the Danielson-Lanczos Lemma to subdivide the initial problem into successively smaller problems, not by factors of 2, but by whatever small prime factors happen to divide N. The larger that the largest prime factor of N is, the worse this method works. If N is prime, then no subdivision is possible, and the user (whether he knows it or not) is taking a *slow* Fourier transform, of order N^2 instead of order $N \log_2 N$. Our advice is to stay clear of such FFT implementations, with perhaps one class of exceptions, the *Winograd Fourier transform algorithms*. Winograd algorithms are in some ways analogous to the base-4 and base-8 FFTs. Winograd has derived highly optimized codings for taking small-N discrete Fourier transforms, e.g., for $N = 2, 3, 4, 5, 7, 8, 11, 13, 16$. The algorithms also use a new and clever way of combining the subfactors. The method involves a reordering of the data both before the hierarchical processing and after it, but it allows a significant reduction in the number of multiplications in the algorithm. For some especially favorable values of N, the Winograd algorithms can be significantly (e.g., up to a factor of 2) faster than the simpler FFT algorithms of the nearest integer power of 2. This advantage in speed, however, must be weighed against the considerably more complicated data indexing involved in these transforms, and the fact that the Winograd transform cannot be done "in place."

Finally, an interesting class of transforms for doing convolutions quickly are number theoretic transforms. These schemes replace floating point arithmetic with integer arithmetic modulo some large prime $N+1$, and the N^{th} root of 1 by the modulo arithmetic equivalent. Strictly speaking, these are not *Fourier* transforms at all, but the properties are quite similar and computational speed can be *far* superior. On the other hand, their use is somewhat restricted to quantities like correlations and convolutions since the transform itself is not easily interpretable as a "frequency" spectrum.

REFERENCES AND FURTHER READING:

Nussbaumer, H.J. 1982, *Fast Fourier Transform and Convolution Algorithms* (New York: Springer-Verlag).

Elliott, D.F., and Rao, K.R. 1982, *Fast Transforms: Algorithms, Analyses, Applications* (New York: Academic Press).

414

Brigham, E. Oran. 1974, *The Fast Fourier Transform* (Englewood Cliffs, N.J.: Prentice-Hall).

Bloomfield, P. 1976, *Fourier Analysis of Time Series – An Introduction* (New York: Wiley).

Beauchamp, K.G. 1975, *Walsh Functions and Their Applications* (New York: Academic Press) [a non-Fourier transform of recent interest].

12.3 FFT of Real Functions, Sine and Cosine Transforms

It happens frequently that the data whose FFT is desired consist of real-valued samples f_j, $j = 0 \dots N-1$. To use four1, we put these into a complex array with all imaginary parts set to zero. The resulting transform F_n, $n = 0 \dots N-1$ satisfies $F_{N-n}{}^* = F_n$. Since this complex-valued array has real values for F_0 and $F_{N/2}$, and $(N/2)-1$ other independent values $F_1 \dots F_{N/2-1}$, it has the same $2(N/2-1)+2 = N$ "degrees of freedom" as the original, real data set. However, the use of the full complex FFT algorithm for real data is inefficient, both in execution time and in storage required. You would think that there is a better way.

There are *two* better ways. The first is "mass production": Pack two separate real functions into the input array in such a way that their individual transforms can be separated from the result. This is implemented in the program twofft below. This may remind you of a one-cent sale, at which you are coerced to purchase two of an item when you only need one. However, remember that for correlations and convolutions the Fourier transforms of two functions are involved, and this is a handy way to do them both at once. The second method is to pack the real input array cleverly, without extra zeros, into a complex array of half its length. One then performs a complex FFT on this shorter length; the trick is then to get the required answer out of the result. This is done in the program realft below.

Transform of Two Real Functions Simultaneously

First we show how to exploit the symmetry of the transform F_n to handle two real functions at once: Since the input data f_j are real, the components of the discrete Fourier transform satisfy

$$F_{N-n} = (F_n)^* \tag{12.3.1}$$

where the asterisk denotes complex conjugation. By the same token, the discrete Fourier transform of a purely imaginary set of g_j's has the opposite symmetry.

$$G_{N-n} = -(G_n)^* \tag{12.3.2}$$

415

Therefore we can take the discrete Fourier transform of two real functions each of length N simultaneously by packing the two data arrays as the real and imaginary parts respectively of the complex input array of four1. Then the resulting transform array can be unpacked into two complex arrays with the aid of the two symmetries. Routine twofft works out these ideas.

```
void twofft(data1,data2,fft1,fft2,n)
float data1[],data2[],fft1[],fft2[];
int n;
```

Given two real input arrays data1[1..n] and data2[1..n], this routine calls four1 and returns two complex output arrays, fft1 and fft2, each of complex length n (i.e. real dimensions [1..2n]), which contain the discrete Fourier transforms of the respective datas. n MUST be an integer power of 2.

```
{
	int nn3,nn2,jj,j;
	float rep,rem,aip,aim;
	void four1();

	nn3=1+(nn2=2+n+n);
	for (j=1,jj=2;j<=n;j++,jj+=2) {
		fft1[jj-1]=data1[j];          Pack the two real arrays into one complex array.
		fft1[jj]=data2[j];
	}
	four1(fft1,n,1);                  Transform the complex array.
	fft2[1]=fft1[2];
	fft1[2]=fft2[2]=0.0;
	for (j=3;j<=n+1;j+=2) {
		rep=0.5*(fft1[j]+fft1[nn2-j]);     Use symmetries to separate the two transforms.
		rem=0.5*(fft1[j]-fft1[nn2-j]);
		aip=0.5*(fft1[j+1]+fft1[nn3-j]);
		aim=0.5*(fft1[j+1]-fft1[nn3-j]);
		fft1[j]=rep;                       Ship them out in two complex arrays.
		fft1[j+1]=aim;
		fft1[nn2-j]=rep;
		fft1[nn3-j] = -aim;
		fft2[j]=aip;
		fft2[j+1] = -rem;
		fft2[nn2-j]=aip;
		fft2[nn3-j]=rem;
	}
}
```

What about the reverse process? Suppose you have two complex transform arrays, each of which has the symmetry (12.3.1), so that you know that the inverses of both transforms are real functions. Can you invert both in a single FFT? This is even easier than the other direction. Use the fact that the FFT is linear and form the sum of the first transform plus i times the second. Invert using four1 with isign=-1. The real and imaginary parts of the resulting complex array are the two desired real functions.

FFT of Single Real Function

To implement the second method, which allows us to perform the FFT of a *single* real function without redundancy, we split the data set in half, thereby forming two real arrays of half the size. We can apply the program above to these two, but of course the result will not be the transform of the original data. It will be a schizophrenic set of two transforms each of which has half of the information we need. Fortunately, this is schizophrenia of a treatable form. It works like this:

The right way to split the original data is to take the even-numbered f_j as one data set, and the odd-numbered f_j as the other. The beauty of this is that we can take the original real array and treat it as a complex array h_j of half the length. The first data set is the real part of this array, and the second is the imaginary part, as prescribed for `twofft`. No repacking is required. In other words $h_j = f_{2j} + i f_{2j+1}$, $j = 0 \ldots N/2 - 1$. We submit this to `four1`, and it will give back a complex array $H_n = F_n^e + i F_n^o$, $n = 0 \ldots N/2 - 1$ with

$$F_n^e = \sum_{k=0}^{N/2-1} f_{2k}\, e^{2\pi i k n/(N/2)}$$
$$F_n^o = \sum_{k=0}^{N/2-1} f_{2k+1}\, e^{2\pi i k n/(N/2)} \qquad (12.3.3)$$

The discussion of program `twofft` tells you how to separate the two transforms F_n^e and F_n^o out of H_n. How do you work them into the transform F_n of the original data set f_j? We recommend a quick glance back at equation (12.2.3):

$$F_n = F_n^e + e^{2\pi i n/N} F_n^o \qquad n = 0 \ldots N-1 \qquad (12.3.4)$$

Expressed directly in terms of the transform H_n of our real (masquerading as complex) data set, the result is

$$F_n = \frac{1}{2}(H_n + H_{N/2-n}{}^*) - \frac{i}{2}(H_n - H_{N/2-n}{}^*)e^{2\pi i n/N} \qquad n = 0, \ldots, N-1 \qquad (12.3.5)$$

A few remarks:

- Since $F_{N-n}{}^* = F_n$ there is no point in saving the entire spectrum. The positive frequency half is sufficient and can be stored in the same array as the original data. The operation can, in fact, be done in place.
- Even so, we need values H_n, $n = 0 \ldots N/2$ whereas `four1` gives only the values $n = 0 \ldots N/2 - 1$. Symmetry to the rescue, $H_{N/2} = H_0$.
- The values F_0 and $F_{N/2}$ are real and independent. In order to actually get the entire F_n in the original array space, it is convenient to put $F_{N/2}$ into the imaginary part of F_0.
- Despite its complicated form, the process above is invertible. First peel $F_{N/2}$ out of F_0. Then construct

$$F_n^e = \frac{1}{2}(F_n + F_{N/2-n}^*)$$
$$F_n^o = \frac{1}{2}e^{-2\pi i n/N}(F_n - F_{N/2-n}^*) \qquad n = 0 \ldots N/2 - 1 \qquad (12.3.6)$$

and use `four1` to find the inverse transform of $H_n = F_n^{(1)} + i F_n^{(2)}$. Surprisingly, the actual algebraic steps are virtually identical to those of the forward transform. Here is a representation of what we have said:

```
#include <math.h>

void realft(data,n,isign)
float data[];
int n,isign;
```

Calculates the Fourier Transform of a set of 2n real-valued data points. Replaces this data (which is stored in array data[1..2n]) by the positive frequency half of its complex Fourier Transform. The real-valued first and last components of the complex transform are returned as elements data[1] and data[2] respectively. n must be a power of 2. This routine also calculates the inverse transform of a complex data array if it is the transform of real data. (Result in this case must be multiplied by 1/n.)

```
{
    int i,i1,i2,i3,i4,n2p3;
    float c1=0.5,c2,h1r,h1i,h2r,h2i;
    double wr,wi,wpr,wpi,wtemp,theta;      Double precision for the trigonometric recur-
    void four1();                              rences.

    theta=3.141592653589793/(double) n;    Initialize the recurrence.
    if (isign == 1) {
        c2 = -0.5;
        four1(data,n,1);                   The forward transform is here.
    } else {
        c2=0.5;                            Otherwise set up for an inverse transform.
        theta = -theta;
    }
    wtemp=sin(0.5*theta);
    wpr = -2.0*wtemp*wtemp;
    wpi=sin(theta);
    wr=1.0+wpr;
    wi=wpi;
    n2p3=2*n+3;
    for (i=2;i<=n/2;i++) {                 Case i=1 done separately below.
        i4=1+(i3=n2p3-(i2=1+(i1=i+i-1)));
        h1r=c1*(data[i1]+data[i3]);        The two separate transforms are separated
        h1i=c1*(data[i2]-data[i4]);            out of data.
        h2r = -c2*(data[i2]+data[i4]);
        h2i=c2*(data[i1]-data[i3]);
        data[i1]=h1r+wr*h2r-wi*h2i;        Here they are recombined to form the true
        data[i2]=h1i+wr*h2i+wi*h2r;            transform of the original real data.
        data[i3]=h1r-wr*h2r+wi*h2i;
        data[i4] = -h1i+wr*h2i+wi*h2r;
        wr=(wtemp=wr)*wpr-wi*wpi+wr;       The recurrence.
        wi=wi*wpr+wtemp*wpi+wi;
    }
    if (isign == 1) {
```

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☑ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

19

McGraw-Hill

A Division of The McGraw-Hill Companies

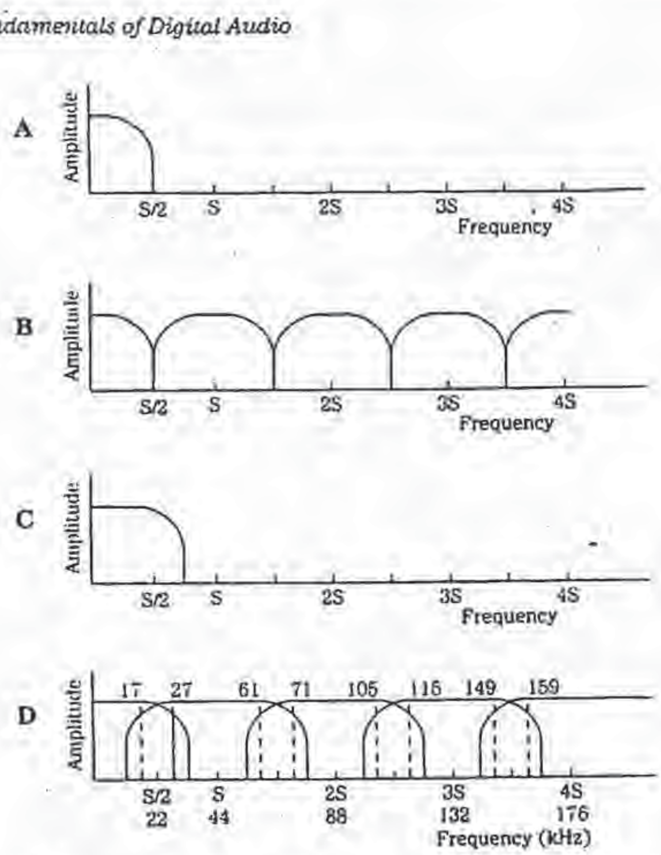
pbk 1 2 3 4 5 6 7 8 9 FGR/FGR 9 0 0 9 8 7 6 5
hc 1 2 3 4 5 6 7 8 9 FGR/FGR 9 0 0 9 8 7 6 5

Library of Congress Cataloging-in-Publication Data
Pohlmann, Ken C.
Principles of digital audio / by Ken C. Pohlmann. — 3rd ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-07-050468-7 ISBN 0-07-050469-5 (pbk.)
1. Sound—Recording and reproducing—Digital techniques.
I. Title.
TK7881.4.P63 1995
621.389'3—dc20
95-17259
CIP

McGraw-Hill books are available at special quantity discounts to use as premiums and sales promotions, or for use in corporate training programs. For more information, please write to the Director of Special Sales, McGraw-Hill, 11 West 19th Street, New York, NY 10011. Or contact your local bookstore.

Acquisitions Editor: Steve Chapman
Editorial team: Joanne Slike, Executive Editor
Andrew Yoder, Supervising Editor
B.J. Peterson, Book Editor
Production team: Katherine G. Brown, Director
Janice Ridenour, Computer Artist
Jeffrey Miles Hall, Computer Artist
Wanda S. Ditch, Desktop Operator
Nancy K. Mickley, Proofreading
Joann Woy, Indexer
Design team: Jaclyn J. Boone, Designer
Katherine Lukaszewicz, Associate Designer

EL1
0504695



2-5 A spectral view of correct sampling, and aliasing. A. An input signal bandlimited to the Nyquist frequency. B. Upon reconstruction, images are contained within multiples of the Nyquist frequency. C. An input signal that is not bandlimited to the Nyquist frequency. D. Upon reconstruction, images are not contained within multiples of the Nyquist frequency; this spectral overlap is aliasing; a 27-kHz signal will alias in a 44-kHz sampler.

aliases at 9 kHz; and the eighth harmonic at 40 kHz aliases at 4 kHz, just below the fundamental.

Alias Prevention

In practice, the problem of aliasing can be overcome. In fact, in a well-designed digital recording system, aliasing does not occur. The solution is straightforward: the input signal is bandlimited with a sharp lowpass filter (anti-aliasing filter) designed to provide significant attenuation at the Nyquist frequency, to ensure that the sampled signal never exceeds the Nyquist frequency. An ideal filter would have a "brick-

wall" characteristic with instantaneous and infinite attenuation in th
However, in practice, the filter cannot achieve this. Rather, it is designed
sition band in which attenuation is achieved over a steeply sloping chan
addition, the filter provides attenuation to the limits of the amplitude r
the system. This ensures that the system meets the demands of the sampling theorem; thus, aliasing cannot occur.

It is critical to observe sampling theory, and lowpass filter the input signal in a digitization system. If aliasing is allowed to occur, there is no technique that can remove the aliased frequencies from the original audio bandwidth. Extremely low level aliasing can occur after the anti-aliasing filter, because of quantization error. A noise signal called dither is used to alleviate this distortion.

Quantization

A measurement of a varying event is meaningful only if both the time and the value of the measurement are stored. Sampling represents the time of the measurement, and quantization represents the value of the measurement, or in the case of audio, the amplitude of the waveform at sample time. Sampling and quantization are thus the fundamental components of digitization, and together can characterize an acoustic event. Both sampling and quantization become variables that determine, respectively, the bandwidth and resolution of the characterization. An analog waveform can be represented by a series of pulses; the amplitude of each pulse yields a number that represents the analog value at that instant. With quantization, as with any analog measurement, accuracy is limited by the system's resolution. Because of finite word length, a digital audio system's resolution is limited, and a measuring error is introduced. This error is often similar to the noise in an analog audio system; however, perceptually, it is more intrusive because its character varies with signal amplitude.

With uniform quantization, an analog signal amplitude is mapped into a number of quanta of equal height. The infinite number of amplitude points on the analog waveform must be quantized by the finite number of quanta levels; this introduces an error. A high-quality representation requires a large number of levels; for example, a high-quality music signal might require 65,536 amplitude levels or more. However, only a few levels can still carry information content; for example, two amplitude levels can (barely) convey intelligible speech.

Consider two voltmeters, one analog and one digital, each measuring the voltage corresponding to an input signal. Given a good meter face and a sharp eye, we might read the analog needle at 1.27 V (volts). A digital meter with only two digits might read 1.3 V. A three-digit meter might read 1.27 V, and a four-digit meter might read 1.274 V. Both the analog and digital measurements contain error. The error in the analog meter is caused by the ballistics of the mechanism and the difficulty in reading the meter. Even under ideal conditions, the resolution of any analog measurement is limited by the measuring device's own noise.

With the digital meter, the nature of the error is different. Accuracy is limited by the resolution of the meter—that is, by the number of digits displayed. The more digits, the greater the accuracy, but the last digit will round off relative to the actual value; for example, 1.27 would be rounded off to 1.3. In the best case, the last digit would be completely accurate; for example, a voltage of exactly 1.3000 would be shown as 1.3. In the worst case, the rounded off digit will be one-half interval away; for example, 1.250 would be rounded off to 1.2 or 1.3. If a binary system is used for the measurement, we say that the error resolution of the system is one-half the LSB (least significant bit). For both analog and digital systems, the problem of measuring an analog phenomenon such as amplitude leads to error. As far as voltmeters are concerned, a digital readout is an inherently more robust measurement. We gain more information about an analog event when it is characterized in terms of digital data. Today, an analog voltmeter is about as useful as a slide rule.

Quantization is thus the technique of measuring an analog event to form a numerical value. A digital system uses a binary number system. The number of possible values is determined by the length of the binary data word—that is, the number of bits available to form the representation. Just as the number of digits in a digital voltmeter determines resolution, the number of bits in a digital audio recorder also determines resolution. In practice, resolution is primarily influenced by the quality of the A/D (analog-to-digital) converter.

Sampling of a bandlimited signal is theoretically a lossless process, but choosing the amplitude value at sample time certainly is not. Any choice of scales or codes shows that digitization can never completely encode a continuous analog function. An analog waveform has an infinite number of amplitude values, but a quantizer has a finite number of intervals. All the analog values between two intervals can only be represented by the single number assigned to that interval. Thus, the quantized value is only an approximation of the actual.

Signal-to-Error Ratio

With a binary number system, the word length determines the number of quantizing intervals available; this can be computed by raising the word length to the power of 2. In other words, an n-bit word would yield 2^n quantization levels. The number of levels determined by the first $n = 1$ to 24 bits are listed in Table 2-1. For example, an 8-bit word provides $2^8 = 256$ intervals and a 16-bit word provides $2^{16} = 65{,}536$ intervals. Note that each time a bit is added to the word length, the number of levels doubles. The more bits, the better the approximation; but as noted, there is always an error associated with quantization because the finite number of amplitude levels coded in the binary word can never completely accommodate an infinite number of analog amplitudes.

It is difficult to appreciate the accuracy achieved by a 16-bit measurement. An analogy might help: if sheets of typing paper were stacked to a height of 22 feet, a single sheet of paper would represent one quantization level in a 16-bit system. Longer word lengths are even more impressive. In a 20-bit system, the stack would reach 349 feet. In a 24-bit system, the stack would tower 5592 feet in height—over a mile high. The quantizer could measure that mile to an accuracy equal to the thickness of a piece of paper. If a single page was removed, the least significant bit would change from 1 to 0. Looked at in another way, if the distance between New York and Los Angeles were measured with 24-bit accuracy, the measurement would be accu-

2^1	= 2
2^2	= 4
2^3	= 8
2^4	= 16
2^5	= 32
2^6	= 64
2^7	= 128
2^8	= 256
2^9	= 512
2^{10}	= 1024
2^{11}	= 2048
2^{12}	= 4096
2^{13}	= 8192
2^{14}	= 16384
2^{15}	= 32768
2^{16}	= 65536
2^{17}	= 131072
2^{18}	= 262144
2^{19}	= 524288
2^{20}	= 1048576
2^{21}	= 2097152
2^{22}	= 4194304
2^{23}	= 8388608
2^{24}	= 16777216

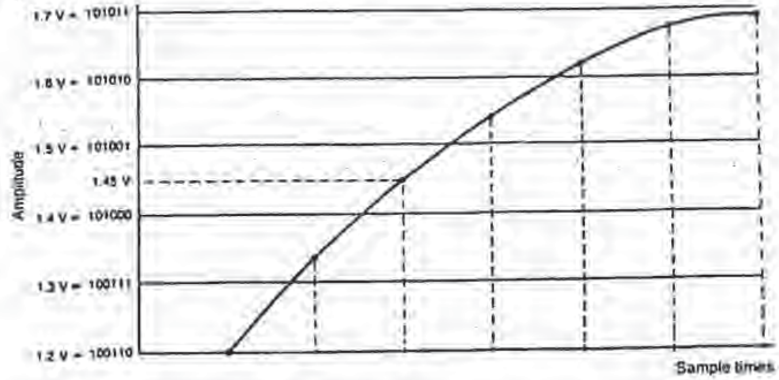
Table 2-1. Number (N) of quantization intervals in a binary word is $N = 2n$ where n is the number of bits in the word.

rate to within 9 inches. A high-quality digital audio system thus requires components with similar tolerances—not a trivial feat.

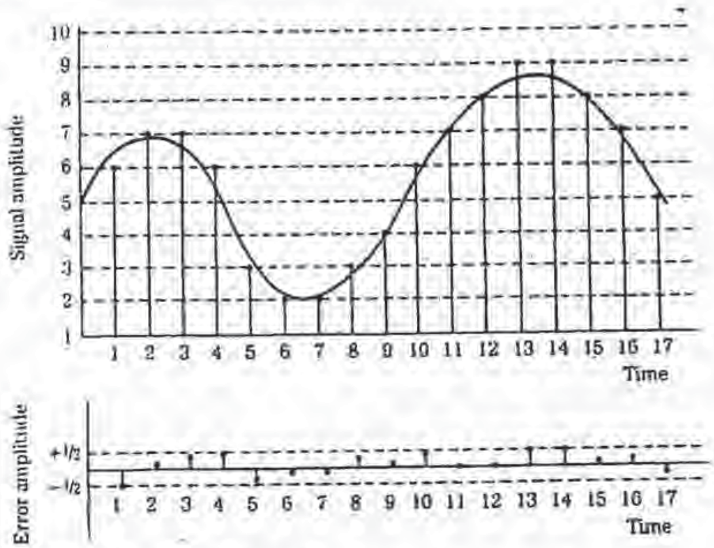
At some point, the quantizing error approaches inaudibility. Most manufacturers have agreed that 16 to 20 bits provide an adequate representation; however, that doesn't rule out longer data words or the use of other signal processing to optimize quantization and thus reduce quantization error level.

Word length determines the resolution of a digitization system and hence provides an important specification for evaluating system performance. Sometimes the quantized interval will be exactly at the analog value; usually it will not be quite exact. At worst, the analog level will be one-half interval away—that is, there is an error of half the least significant bit of the quantization word. For example, consider Fig. 2-6. Suppose the binary word 101000 corresponds to the analog interval of 1.4 V, 101001 corresponds to 1.5 V, and the analog value at sample time is unfortunately 1.45 V. Because 101000½ is not available, you must round up to 101001 or down to 101000. Either way, there will be an error with a magnitude of one-half of an interval.

Quantization error is the difference between the actual analog value at sample time and the selected quantization interval value. At sample time, the amplitude value is rounded to the nearest quantization interval, as shown in Fig. 2-7. At best (sample points 11 and 12 in the figure), the waveform coincides with quantization intervals. At worst (sample point 1 in the figure), the waveform is exactly between two intervals. Quantization error is thus limited to a range between $+Q/2$ and $-Q/2$, where Q is one quantization interval. Note that this selection process, of one level or another, is the basic mechanism of quantization, and occurs for all samples in a digi-



2-6 Quantization error is limited to one-half LSB.



2-7 Quantization error at sample times.

tal system. This error results in distortion that is present for any amplitude audio signal. When the signal is large, the distortion is relatively small and masked. However, when the signal is small, the distortion is relatively large and might be audible.

In characterizing digital hardware performance, we can determine the ratio of the maximum expressible signal amplitude to the maximum quantization error; this determines the S/E (signal-to-error) ratio of the system. The signal-to-error ratio of a digital system is closely akin, but not identical to the S/N (signal-to-noise) ratio of an analog system. The S/E relationship can be derived using a ratio of signal-to-ι voltage levels.

Consider a quantization system in which n is the number of bits, and N is the number of quantization steps. As noted:

$$N = 2^n$$

where one bit is a sign bit.

Half of these 2^n values will be used to code each part of the bipolar waveform. If Q is the quantizing interval, the peak values of the maximum signal levels are $\pm Q2^{n-1}$. Assuming a sinusoidal input signal, the maximum rms (root mean square) signal S_{rms} is:

$$S_{rms} = \frac{Q2^{n-1}}{(2)^{1/2}}$$

The energy in the quantization error also can be determined. When the input signal has high amplitude and wide spectrum, the quantization error is statistically independent and uniformly distributed between the $+Q/2$ and $-Q/2$ limits, and 0 elsewhere, where Q is one quantization interval. This dictates a uniform probability density function with amplitude of $1/Q$; the error is random from sample to sample; the error spectrum is flat. Ignoring error outside the signal band, the rms quantization error E_{rms} can be found by summing (integrating) the product of the error and its probability:

$$E_{rms} = \left[\int_{-\infty}^{\infty} e^2 p(e)de\right]^{1/2} = \left[\frac{1}{Q}\int_{-\frac{Q}{2}}^{\frac{Q}{2}} e^2 de\right]^{1/2} = \left[\frac{Q^2}{12}\right]^{1/2} = \frac{Q}{(12)^{1/2}}$$

The power ratio determining the signal to quantization error is:

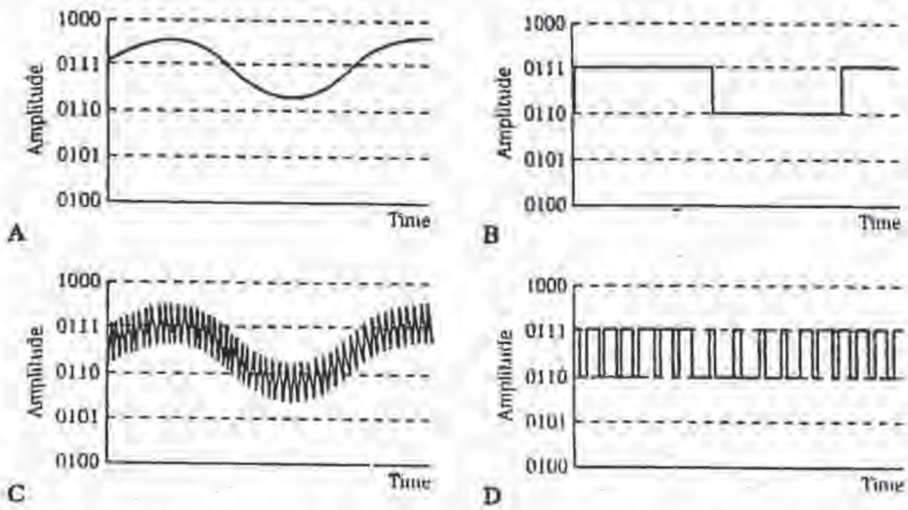
$$\frac{S}{E} = \left[\frac{S_{rms}}{E_{rms}}\right]^2 = \frac{\left[\frac{Q2^{n-1}}{(2)^{1/2}}\right]^2}{\left[\frac{Q}{(12)^{1/2}}\right]^2} = \frac{3}{2}(2^{2n})$$

Expressing this ratio in decibels:

$$\frac{S}{E}\text{ (dB)} = 10\log\left[\frac{3}{2}(2^{2n})\right] = 20\log\left[\left(\frac{3}{2}\right)^{1/2}\left(2^n\right)\right]$$
$$= 6.02n + 1.76$$

Using this approximation for signal-to-error ratio, we observe that ideal 16-bit quantization yields an S/E ratio of about 98 dB, but 15-bit quantization is inferior at 92 dB. In other words, each additional bit reduces the quantization noise by 6 dB, or a factor of two. Longer word lengths increase the data signal bandwidth required to convey the signal. However, the signal-to-quantization noise power ratio increases exponentially with data signal bandwidth. This is an efficient relationship that approaches the theoretical maximum, and it is a hallmark of coded systems such as PCM (pulse-code modulation) described in chapter 3. The figure of 1.76 is based on the statistics (peak-to-rms ratio) of a sinusoidal waveform; it will differ if the signal peak-to-rms ratio differs from that of a sinusoid.

It also is important to note that this result assumes that the quantization error is uniformly distributed, and quantization is accurate enough to prevent signal correlation in the error waveform. This is generally true for high amplitude complex audio



2-8 Dither is used to alleviate the effects of quantization error. A. An undithered input signal with amplitude on the order of one LSB. B. Quantization results in a coarse coding over two levels. C. A dithered input signal. D. Quantization yields a PWM waveform that codes information below the LSB.

quantized together, and this randomizes the error. This linearizes the quantization process. This technique is known as nonsubtractive dither because the dither signal is permanently added to the audio signal; the total error is not statistically independent of the audio signal, and errors are not independent sample to sample. However, nonsubtractive dither does manipulate the statistical properties of the quantizer, statistically rendering conditional moments of the total error independent of the input, effectively decorrelating the quantization error of the samples from the signal, and from each other. The power spectrum of the total error signal can be made white. Subtractive dithering, in which the dither signal is removed after requantization, theoretically provides total error statistical independence, but is more difficult to implement.

John Vanderkooy and Stanley Lipshitz have demonstrated the benefit of dither with a 1-kHz sinewave with a peak-to-peak amplitude of one LSB, as shown in Fig. 2-9. Without dither, a square wave is output from the digital-to-analog converter. When Gaussian dither with an rms amplitude of ½ LSB is added to the original signal, a pulse-width-modulated waveform results. The encoded sinewave is revealed when the signal is averaged over many periods. A sinewave emerges from the PWM output signal. The averaging illustrates how the ear responds in its perception of acoustic signals; that is, the ear is a lowpass filter that averages any signal. In this case, a noisy sinewave is heard, rather than a square wave.

The ear is quite good at resolving narrow-band signals below the noise floor, because of the averaging properties of the basilar membrane. The ear behaves as a one-

Dither

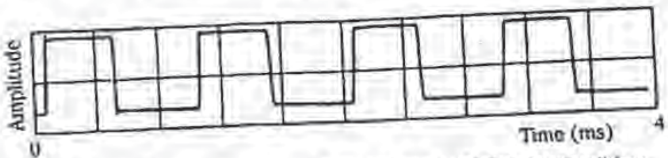
With large amplitude complex signals, there is little correlation between the signal and quantization error; thus the error is random and perceptually similar to analog white noise. With low-level signals, the character of the error changes as it becomes correlated to the signal, and potentially audible distortion results. A digitization system must suppress any audible qualities of its quantization error. Obviously, the number of bits in the quantizing word can be increased, resulting in a decrease in error amplitude of 6 dB per additional bit. This is uneconomical, and many bits are needed to satisfactorily reduce the audibility of quantization error.

Dither is a far more efficient technique. With dither, a small amount of noise is added to the audio signal prior to sampling to linearize the quantization process. Essentially, with dither the audio signal is made to shift with respect to quantization levels. The averaging process smooths the effect of incremental quantization levels and decorrelates the error from the signal. This randomizes the effects of the quantization error to the point of total elimination. However, although it greatly reduces distortion, dither adds some noise to the output signal.

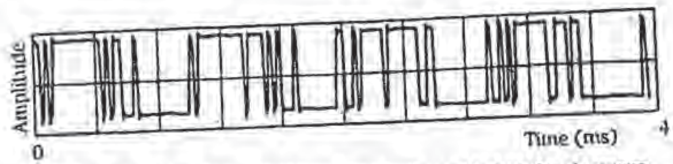
Dither does not mask quantization error; rather, it allows the digital system to encode amplitudes smaller than the least significant bit, in much the same way that an analog system can retain signals below its noise floor. A properly dithered digital system far exceeds the signal to noise performance of an analog system. On the other hand, an undithered digital system can be inferior to an analog system, particularly under low-level signal conditions. A high-quality digital audio system demands dithering prior to quantization at the A/D converter. In a very conceptual sense, dither is similar to high frequency bias in an analog magnetic tape recorder. In addition, digital computations should be dithered prior to requantization at a D/A converter.

Consider the case of an input audio signal with amplitude on the order of one quantization interval. It would either move within the interval, resulting in a dc (direct current) signal, or move back and forth across the interval threshold, resulting in an output square wave, as shown in Fig. 2-8A and B. The square wave demonstrates that quantization ultimately acts as a hard limiter; in other words, severe distortion takes place. The effect is quite different when a dither noise signal is added to the audio signal. The result shown in Fig. 2-8C and D is a pulse signal that preserves the low-level information of the audio signal. The quantized signal switches up and down as the dithered input varies, tracking the average value of the input signal. This information is encoded in the varying width of the quantized signal pulses. This kind of information storage is known as pulse-width modulation, and it accurately preserves the input signal waveform. The average value of the quantized signal moves continuously between two levels, alleviating the effects of quantization error. Similarly, analog noise would be coded as a binary noise signal; values of 0 and 1 would appear in the LSB in each sampling period, with the signal retaining its white spectrum. The perceptual result is the original signal with added noise—a more desirable result than a quantized square wave.

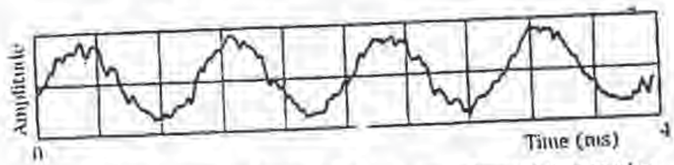
Mathematically, with dither, quantization error is no longer a deterministic function of the input signal, ~~but~~ rather becomes a zero-mean random variable. In other words, rather than quantizing only the input signal, the dither noise and signal are



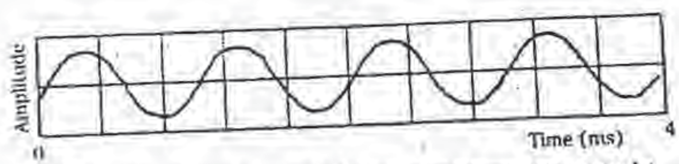
A A 1-kHz sinewave with amplitude of one-half LSB without dither produces a square wave.



B Dither of one-third LSB rms amplitude is added to the sinewave before quantization, resulting in a PWM waveform.



C Modulation carries the encoded sinewave information, as can be seen after 32 averagings.



D Modulation carries the encoded sinewave information, as can be seen after 960 averagings.

2-9 Dither permits encoding of information below the least significant bit.
Vanderkooy and Lipshitz

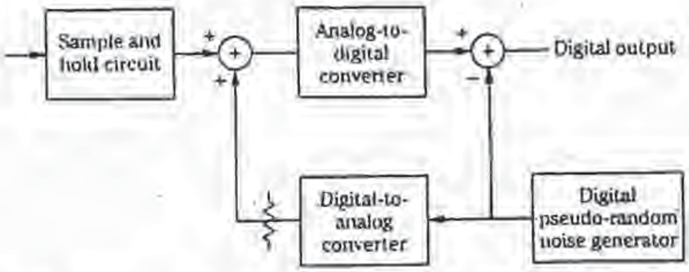
third octave filter with a narrow bandwidth; the quantization error, which is given a white noise character by dither, is averaged by the ear, and the original narrow-band sinewave is heard without distortion. In other words, dither changes the digital nature of the quantization error into a white noise, and the ear can then resolve signals with levels well below one quantization level.

This conclusion is an important one. With dither, the resolution of a digitization system is far below the least significant bit; theoretically, there is no limit to low-level resolution. By encoding the audio signal with dither to modulate the quantize[d sig]nal, that information can be recovered, even though it is smaller than the smallest quantizer interval. Furthermore, dither can eliminate distortion caused by quantization, by reducing those artifacts to white noise. Proof of this is shown in Fig. 2-10, illustrating a computer simulation performed by John Vanderkooy, Robert Wannamaker, and Stanley Lipshitz. The figure shows a 1-kHz sinewave of 4.0 LSB peak-to-peak amplitude. The first column shows the signal without dither. The second column shows the same signal with triangular pdf (probability density function) dither (explained in the following paragraphs) of 2.0 LSB peak-to-peak amplitude. In both cases, the first row shows the input signal. The second row shows the output signal. The third row shows the total quantization error signal. The fourth row shows the power spectrum of the output signal (this is estimated from sixty 50% overlapping Hanning windowed 512-point records at 44.1 kHz). The undithered output signal (D) suffers from harmonic distortion, visible at multiples of the input frequency, as well as inharmonic distortion from aliasing. The error signal (G) of the dithered signal shows artifacts of the input signal; thus, it is not statistically independent. However, surprisingly, this error signal sounds like white noise (although it clearly does not look like white noise) and the output signal sounds like a sinewave with noise. This is supported by the power spectrum (H) showing that the signal is quite free of signal-dependent artifacts, with a white noise floor. However, we can see that dither increases the noise floor of the output signal.

Types of Dither

There are several types of dither signals, generally differentiated by their pdf (probability density function). Given a random signal with a continuum of possible values, the integral of the probability density function describes the probability of the values over an interval. The probability that the signal falls between the interval is the area under the function. For example, the probability might be constant over an interval, or it might vary. For audio applications, interest has focused on three dither signals: Gaussian pdf, rectangular (or uniform) pdf, and triangular pdf, as shown in Fig. 2-11. For example, we might speak of a statistically independent, white dither signal with a triangular pdf having a level or width of 2 LSB. Generally, dither signals have a white spectrum; however, the spectrum can be shaped by correlating successive dither samples without modifying the pdf; for example, a highpass triangular pdf dither signal could be created. All three dither types are effective at linearizing the transfer characteristics of quantization, but differ in their results. Although rectangular and triangular pdf dither signals add less overall noise to the signal, Gaussian dither is easier to implement in the analog domain.

Rectangular and triangular pdf dither of constant and precise amplitude are costly to generate in the analog domain; for example, the signal from a pseudo-random number generator could be applied to a D/A (digital-to-analog) converter to create a rectangular pdf signal. Therefore, designers often employ Gaussian noise as dither prior to A/D conversion. Gaussian dither is easy to generate with common



2-13 An example of a subtractive digital dither circuit using a pseudo-random number generator.

converter, and the other subtracting it at the D/A converter. Alternatively, in an autodither system, the audio signal itself could be randomized to create an added dither at the A/D converter, then re-created at the D/A converter and subtracted from the audio signal to restore dynamic range.

Digital dither must be used to decrease round-off error when signal manipulation takes place in the digital domain. For example, the truncation associated with multiplication can cause objectionable error, as described in chapter 15.

For the sake of completeness, Jim MacArthur has pointed out that one of the earliest uses of dither came in World War II; bombers used mechanical computers to perform navigation and bomb trajectory calculations. Curiously, these computers (boxes filled with hundreds of gears and cogs) performed more accurately when flying on board the aircraft, and less well on ground. Engineers realized that the vibration from the aircraft reduced the error from sticky moving parts—instead of moving in short jerks, they moved more continuously. Small vibrating motors were built into the computers, and their vibration was called dither from the Middle English verb "didderen," meaning "to tremble." Today, when you tap a mechanical meter to increase its accuracy, you are applying dither, and modern dictionaries define dither as "a highly nervous, confused, or agitated state." In minute quantities, dither successfully makes a digitization system a little more analog in the good sense of the word.

Conclusion

Sampling and quantizing are the two fundamental criteria for a digitization system. The sampling frequency determines signal bandlimiting and thus frequency response. Although complex, sampling is based on well-understood principles; the cornerstone of discrete time sampling yields completely predictable results. Aliasing occurs when the sampling theorem is not observed. Quantization determines the dynamic range of the system, measured by the signal-to-error ratio. Although bandlimited sampling is a lossless process, quantization is one of approximation. Quantization artifacts can severely affect the performance of a system. However, dither can elim-

Although a perfect error-correction system is theoretically possible, in which every error is detected and corrected, such a system would create an unreasonably high data overhead because of the amount of redundant data required to accomplish it. Thus, an efficient audio error-correction system should aim to provide a low audible error rate after correction and concealment, while minimizing the amount of redundant data and data processing required for successful operation. An error-correction system comprises three operations:

1. Error detection uses redundancy to permit data to be checked for validity
2. Error correction uses redundancy to replace erroneous data with newly calculated valid data
3. In the event of large errors or insufficient data for correction, error concealment techniques substitute approximately correct data for invalid data

In the worst case, when not even error concealment is possible, digital audio systems mute the output signal rather than let the output circuitry attempt to decode severely incorrect data, and produce severely incorrect sounds.

Error Detection

All error-detection and correction techniques are based on the redundancy of data. The data is said to be redundant because it is entirely derived from existing data, and thus conveys no additional information. In general, the greater the likelihood of errors, the greater the redundancy required. Information systems rely heavily on redundancy to achieve reliable communication; for example, spoken and written language contains redundancy. If a garbled telegraph message "AJL IS FIRGIVRN. PLEAOE COMW HOME," is received, the message could be recovered. In fact, Claude Shannon estimated that 50% of written English is redundant.

Similarly, redundancy is required for reliable data communication. If a data value alone is generated, transmitted once, and received, there is no absolute way to check its validity at the receiving end. We might examine the data word by word, and question, for example, a word that unexpectedly differs from its neighbors. With digital audio, in which there is some sample correlation from one forty-thousandth of a second to the next, such an algorithm might be reasonable. However, we could not absolutely detect errors, or begin to correct them. Clearly, additional information is required to reliably detect errors in received data. Moreover, such information must originate from the same point as the original data so that it is subject to the same error-creating conditions as the data itself. The task of error detection is to properly code transmitted or stored information, so that when data is lost or made invalid, the presence of the error can be detected.

In an effort to detect errors, the original message could simply be repeated. For example, each data word could be transmitted twice. A conflict between repeated words would reveal that one is in error, but it would be impossible to identify the correct word. If each word was repeated three times, probability would suggest that the two in agreement were correct while the differing third was in error. Yet all three words could agree and all be in error, unknown to us. Given enough repetition, the probability of correctly detecting an error would be high; however, the data over-

CRCC also influences how accurate the CRCC detection must be. The CRCC is typically used as an error pointer to identify the number and extent of errors prior to other error-correction processing.

Error-Correction Codes

With the use of redundant data, it is possible to correct errors that occur during transmission or storage of digital audio data. In the simplest case, data is simply duplicated. For example, instead of writing only one data track to recorded tape, two tracks of identical data could be written. The first track would normally be used for playback, but if an error were detected through parity or other means, data could be taken from the second track. To alleviate the problem of simultaneously erroneous data, redundant samples could be displaced with respect to each other in time.

In addition, channel coding can be used beneficially. For example, three-bit sequences could be coded as 7-bit words, selected from 2^7 possible combinations to be as mutually different as possible. The receiver examines the 7-bit words and compares them to the eight allowed code words. Errors could be detected, and the words changed to the nearest allowed code word, before the code word is decoded to the original three-bit sequence. Four correction bits are required for every 3 data bits; the method can correct a single error in a 7-bit block. This minimum length concept is important in more sophisticated error-correction codes.

Although such simple methods are workable, they are inefficient because of the data overhead they require. A more enlightened approach is that of error correcting codes, which can achieve more reliable transmission or storage with less redundancy. In the same way that redundant data in the form of parity check bits is used for error detection, redundant data is used to form codes for error correction. Digital audio is encoded with related detection and correction algorithms. On playback, errors are identified and corrected by the detection and correction decoder. Coded redundant data is the essence of all correction codes; however, there are many types of codes, different in their designs and functions.

The field of error-correction codes is a highly mathematical one. Many types of codes, developed for different applications, have been developed. In general, two approaches are used: block codes using algebraic methods, and convolutional codes using probabilistic methods. Block codes form a coded message based solely on the message parsed into a data block. In a convolutional code, the coded message is formed from the message present in the encoder at that time as well as previous message data. In many cases, algorithms use a block code in a convolutional structure known as a cross-interleave code. Such codes are used in the DASH and CD formats. 3 approaches

Block Codes

Block error-correction encoders assemble a number of data words to form a block and, operating over that block, generate one or more parity words and append them to the block. During decoding, an algorithm forms a syndrome word that detects errors and, given sufficient redundancy, corrects them. Such algorithms are effective against errors encountered in digital audio applications. Error correction is enhanced by interleaving consecutive words. Block codes base their parity calculations on an entire block of information to form parity words. In addition, parity can be formed from individual words in the block, using 1-bit parity or a cyclic code. In this way, greater redundancy is achieved and correction is improved. For example, CRCC could be used to detect an error, then block parity used to correct the error.

A block code can be conceived as a binary message consolidated into a block, with row and column parity. Any single word error will cause one row and one column to be in error; thus the erroneous data can be corrected. For example, a message might be grouped into four 8-bit words (called symbols). A parity bit is added to each row and a parity word added to each column, as shown in Fig. 5-10. At the decoder, the data is checked for correct parity, and any single symbol error is corrected. In this example, bit parity shows that word three is in error, and word parity is used to correct the symbol. A double word error can be detected, but not corrected. Larger numbers of errors might result in misdetection or miscorrection.

```
Transmitted      Transmitted
data block       single bit parity

00010111         0
01101010         0
10010111         1
11010110         1
--------
00111100 ------------------------ Transmitted parity word

Received         Received
data block       parity bit

00010111         0
01101010         0
11100100         1
11010110         1
--------
00111100 ------------------------ Received parity word
01110011 ------------------------ Parity word calculated from
                                  received data and parity word

Parity calculated
on received data block

      0
      0
      0 -------- Indicates error in word 3
      1

  01110011 ---- Calculated parity word
+ 11100100 ---- Incorrect word 3
  --------
  10010111 ---- Corrected word 3
```

5-10
An example of block parity with row parity bits and column parity word.

Block correction codes use many methods to generate the transmitted code word and its parity; however, they are fundamentally identical in that only information from the block itself is used to generate the code. The extent of the correction capabilities of block correction codes can be simply illustrated with decimal number examples. Given a block of six data words, a seventh parity word can be calculated by adding the six data words. To check for an error, a syndrome is created by comparing (subtracting in the example) the parity (sum) of the received data with the received parity value. If the result is zero, then most probably no error has occurred, as shown in Fig. 5-11A. If one data word is detected and the word is set to zero, a condition called a single erasure, a nonzero syndrome indicates that; furthermore, the erasure value can be obtained from the syndrome, as shown in Fig. 5-11B. If CRCC or 1-bit parity is used, it points out the erroneous word, and the correct value can be calculated using the syndrome, as shown in Fig. 5-11C. Even if detection itself is in error and falsely creates an error pointer, the syndrome yields the correct result, as shown in Fig. 5-11D. Such a block correction code is capable of detecting a one-word error, or making one erasure correction, or correcting one error with a pointer. The correction ability depends on the detection ability of pointers. In this case, unless the error is identified with a pointer, erasure, or CRCC detection, the error cannot be corrected, as shown in Fig. 5-11E.

For enhanced performance, two parity words can be formed to protect the data block. For example, one parity word might be the sum of the data and the second parity word the weighted sum as shown in Fig. 5-12A. If any two words are erroneous and marked with pointers, the code provides correction, as shown in Fig. 5-12B. Similarly, if any two words are marked with erasure, the code can use the two syndromes to correct the data. Unlike the single parity example, this double parity code also can correct any one-word error, even if it is not identified with a pointer, as shown in Fig. 5-12C. This type of error correction is well suited for audio applications.

Cyclic codes, such as CRCC, are a subclass of linear block codes, which can be used for error correction. Special block codes, known as Hamming codes, create syndromes that point to the location of the error. Multiple parity bits are formed for each data word, with unique encoding. For example, three parity check bits (4, 5, and 6) might be added to a 4-bit data word (0, 1, 2, and 3); seven bits are then transmitted. For example, suppose that the three parity bits are uniquely defined as follows: parity bit four is formed from modulo 2 addition of data bits 1, 2, and 3; parity bit 5 is formed from data bits 0, 2, and 3; and parity bit 6 is formed from data bits 0, 1, and 3. Thus, the data word 1100, appended with parity bits 110, is transmitted as the 7-bit code word 1100110. A table of data and parity bits is shown in Fig. 5-13A.

This algorithm for calculating parity bits is summarized in Fig. 5-13B. An error in a received data word can be located by examining which of the parity bits detects an error. The received data must be correctly decoded; therefore, parity check decoding equations must be written. These equations are computationally represented as a parity check matrix H, as shown in Fig. 5-13C. Each row of H represents one of the original encoding equations. By testing the received data against the values in H, the location of the error can be identified. Specifically, a syndrome is calculated from the modulo 2 addition of the parity calculated from the received data and the received parity. An error generates a 1; otherwise a 0 is generated. The resulting error pattern is matched in the H matrix to locate the erroneous bit. For example, if the code word 1100110 is transmitted, but 1000110 is received, the syndromes will detect the error and generate a 101 error pattern. Matching this against the H matrix, we see that it corresponds to the second column; thus, bit 1 is in error, as shown in Fig. 5-13D. This algorithm is a single error correcting code; therefore, it can correctly identify and correct any 1-bit error.

Returning to the design of this particular code, we can observe another of its interesting properties. Referring again to Fig. 5-13A, recall that the seven-bit data words are each comprising four data bits and three parity bits. These seven bits proirst generation copying is permissible, but not second generation mple, a user can digitally copy from CD to a DAT, but a copy-inhibit DAT tape's subcode so that it is impossible to digitally copy from the other DAT tape. However, a SCMS-equipped DAT recorder can ber of digital copies from an original source. SCMS does not affect n any way. SCMS is a fair solution because it allows a user to make a digital copy of purchased software, for example, for compilation of favorite songs, but helps prevent a second party from copying music that was not paid for. On the other hand, SCMS might prohibit the recopying of original recordings, a legitimate use. Use of SCMS is mandated in the U.S. by the Audio Home Recording Act of 1992, as passed by Congress to protect copyrighted works.

The SCMS circuit is found in consumer-grade recorders with S/PDIF (IEC-958 type II) interfaces; it is not present in professional AES3 (IEC-958 type I) interfaces. In particular, SCMS resides in the channel status bits as defined in IEC-958 type II, Amendment No. 1 standard; this data is used to determine whether the data is copyrighted, and whether it is original, or copied. The SCMS circuit first examines the channel status block (see Fig. 10-7) in the incoming digital data to determine whether it is a professional bit stream, or a consumer bit stream. In particular, when byte 0 bit 0 is a 1 the bit stream is assumed to adhere to the AES3 standard; SCMS takes no action. SCMS signals do not appear on AES3 interfaces, and the AES3 standard does not recognize nor carry SCMS information; thus, audio data is not copy-protected, and can be indefinitely copied. When bit 0 is set to 0, the SCMS identifies the data as consumer data. It examines byte 0 bit 2, the copyright or C bit; it is set to 0 when copyright is enabled, and set to 1 when copyright is not enabled. Byte 1 bit 7 (the 15th bit in the block) is the generation or L bit; it is used to indicate the generation of the recording. For most category codes, an L bit of 0 indicates that the transmitted signal is a copy and a 1 means the signal is original. However, the meaning is reversed for laser optical products, and broadcast reception: 0 indicates an original, and 1 indicates a copy. The L bit is thus interpreted by the category code contained in byte 1 bits 0–6 that indicates the type of transmitting device. In the case of the compact disc, because the L bit is not defined in the CD standard (IEC 908), the copy bit designates both copyright and generation. The disc is not copyrighted if the C bit is 0; the disc is copyrighted and original if C is 1; if C alternates between 0 and 1 at a 4–10-Hz rate, the disc is copyrighted for the first generation or higher. Also, because the general category and A/D converter category without copyrighting cannot carry C or L information, these bits are ignored and the receiver sets C for copyright, and L to original.

Generally, the following recording scenario exists when bit 0 is set to 0, indicating a consumer bit stream: When bit C is 1, incoming audio data will be recorded no matter what is written in the category code or L bit, and the new copy can in turn be copied an unlimited number of times. When bit C is 0, the L bit is examined; if the incoming signal is a copy, no recording is permitted. If the incoming signal is original, it will be recorded, but the recording is marked as a copy by setting bits in the recording's subcode; it cannot be copied. When no defined category code is present, one generation of copying is permitted. When there is a defined category code but no copyright information, two generations are permitted. However, different types of equipment respond differently to SCMS. For example, equipment that does not store, decode, or interpret the transmitted data is considered transparent and ignores SCMS flags. Digital mixers, filters, optical disk recorders and tape recorders require different interpretations of SCMS; the general algorithm used to interpret SCMS code is thus rather complicated.

By law, the SCMS circuit must be present in consumer recorders with the S/PDIF or IEC-958 type II interconnection; however, some professional recorders, essentially upgraded consumer models, also contain an SCMS circuit. If recordists use the S/PDIF interface, copy-inhibit flags are sometimes inadvertently set, leading to problems when subsequent copying is needed.

AES11 Digital Audio Reference Signal

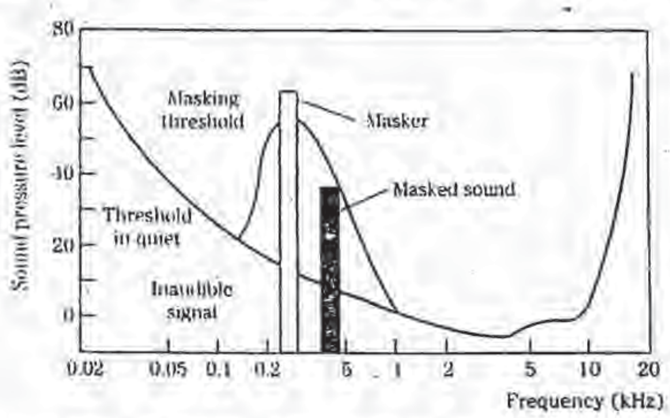
The AES11-1990 standard specifies criteria for synchronization of digital audio equipment in studio operations. It is important for interconnected devices to share a common timing signal so that individual samples are processed simultaneously; timing inaccuracies can lead to increased noise, and even clicks and pops in the audio signal. With a proper reference, transmitters, receivers, and D/A converters can all work in unison. Devices must be synchronized in both frequency and phase, and be SMPTE time synchronous as well. It is relatively easy to achieve frequency synchronization between two sources—they must follow a common clock, and the signals' bit periods must be equal. However, to achieve phase synchronization, the bit edges in the different signals must begin simultaneously.

When connecting one digital audio device to another, the devices must operate at a common sampling frequency, and bits in the sending and received signals must begin simultaneously. These synchronization requirements are relatively easy to achieve. Most digital audio data streams are self-clocking; the receiving circuits read the incoming modulation code, and reference the signal to an internal clock to produce stable data. In some cases, an independent synchronization signal is transmitted. In either case, in simple applications, the receiver can lock to the bit stream's sampling frequency.

However, with numerous devices, it is difficult to obtain frequency and phase synchronization. Different types of devices use different time-bases hence they exhibit noninteger relationships. For example, at 44.1 kHz, a digital audio bit stream will clock 1471.47 samples per NTSC video frame; sample edges align only once every 10 frames. Other data, such as the 192 sample channel status block, creates additional synchronization challenges; in this case, the audio sample clock, channel status, and video frame will align only once every 20 minutes. To achieve synchronization, a common clock with good frequency stability should be distributed through a studio. In addition, external synchronizers are needed to read SMPTE timecode, and provide time synchronization between devices. Figure 10-8 shows an example of synchronization for an audio/video studio; timecode is used to provide general time lock; a master oscillator (using AES11 or video sync) provides a stable clock to ensure frequency lock of primary devices (the analog multitrack is locked via an external synchronizer and synthesizers are not locked). It is important that the timecode reference is different from the frequency lock reference. In addition, most timecode sources are not sufficiently accurate to provide frequency and phase lock references through a studio.

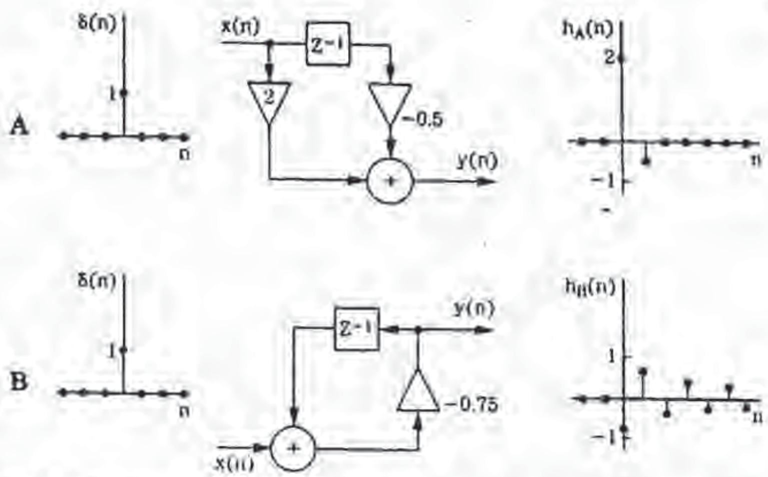
Threshold of Hearing, and Masking

Two fundamental phenomena that govern human hearing are the minimum hearing threshold, and masking, as shown in Fig. 11-6. The threshold of hearing curve describes the minimum level (0 sone) at which the ear can detect a tone at a given frequency. The threshold is referenced to 0 dB at 1 kHz. The ear is most sensitive around 1 to 5 kHz, where we can hear signals several decibels below the 0-dB reference. Generally, two tones of equal power and different frequency will not sound equally loud. Similarly, the audibility of noise and distortion varies according to frequency. Sensitivity decreases at high and low frequencies. For example, a 20-Hz tone would have to be approximately 70 dB louder than a 1-kHz tone to be barely audible. Perceived loudness can be expressed in sones; one sone describes the loudness of a 40 dB SPL sine tone at 1 kHz. A loudness of 2 sones corresponds to 50 dB SPL; similarly, any doubling of loudness in sones results in a 10-dB increase in SPL. For example, 64 sones corresponds to 100 dB SPL. A perceptual coder compares the input signal to the minimum threshold, and discards signals that fall below the threshold, because the ear cannot hear these signals.



11-6 The threshold of hearing describes the softest sounds audible across the human hearing range. A masker tone or noise will raise the threshold of hearing in a local region, creating a masking curve. Masked tones or noise, perhaps otherwise audible, that fall below the masking curve during that time will not be audible.

Amplitude masking occurs when a tone shifts the threshold curve upward in a frequency region surrounding the tone. The masking threshold describes the level where a tone is barely audible. When tones are sounded simultaneously, masking occurs in which louder tones can completely obscure softer tones. In other words, the physical presence of sound certainly does not ensure audibility and conversely can ensure inaudibility of other sound. The strong sound is called the masker and the softer sound is called the maskee. Masking theory argues that the softer tone is just



15-8 LTD systems can be characterized by their impulse responses. A. A simple nonrecursive system and its impulse response. B. A simple recursive system and its impulse response. van den Enden and Verhoeckx

operate with sample numbers, the time of a delay can be obtained by taking nT, where T is the sampling interval. Figure 15-8 shows two examples of simple networks and their impulse responses; as described (see Fig. 15-1B). LTD systems such as these are completely described by the impulse response.

In practice, these elemental operations are performed many times for each sample, in specific configurations depending on the desired result. In this way, algorithms can be devised to perform operations useful to audio processing, such as reverberation, equalization, data compression, limiting, and noise removal. Of course, for real-time operation, all processing for each sample must be completed within one sampling period of 20 µs or so.

Digital Filters

Filtering (or equalization) is important in many audio applications. Analog filters using both passive and active designs shape the signal's frequency response and phase, as described by linear time-invariant differential equations. They describe the system's performance in the time domain. With digital filters, each sample is processed through a transfer function to affect the change in frequency response or phase. Operation is generally described in linear shift-invariant difference equations; they define how the discrete time signal behaves from moment to moment, in the time domain. At an infinitely high sampling rate, these equations would be identical to those used to describe analog filters. Digital filters can be designed from analog filters; such impulse-invariant design is useful for lowpass filters with a cutoff frequency far below the sampling rate. Other filter designs make use of transformations to convert characteristics of an analog filter to a digital filter. These transformations map the frequency range of the analog domain into the digital range, from 0 Hz to the Nyquist frequency.

A digital filter can be represented by a general difference equation:

$$y(n) + b_1 y(n-1) + b_2 y(n-2) + \dots b_N y(n-N) =$$
$$a_0 x(n) + a_1 x(n-1) + a_2 x(n-2) + \dots a_M x(n-M)$$

More efficiently, the equation can be written:

$$y(n) = \sum_{i=0}^{M} a_i x(n-i) - \sum_{i=1}^{N} b_i y(n-i)$$

where x is the input signal, y is the output signal, the constants a_i and b_i are the filter coefficients, and n represents the current sample time, the variable in the filter's equation. A difference equation is used to represent $y(n)$ as a function of the current input, previous inputs, and previous outputs. The filter's order is specified by the maximum time duration (in samples) used to generate the output. For example, the equation:

$$y(n) = x(n) - y(n-2) + 2x(n-2) + x(n-3)$$

is a third-order filter.

To implement a digital filter, the z-transform is applied to the difference equation so that it becomes:

$$Y(z) = \sum_{i=0}^{M} a_i z^{-i} X(z) - \sum_{i=1}^{N} b_i z^{-i} Y(z)$$

where z^{-i} is a unit of delay i in the time domain. Rewriting the equation, the transfer function $H(z)$ can be determined:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{M} a_i z^{-i}}{(1 + \sum_{i=1}^{N} b_i z^{-i})}$$

As noted, the transfer function can be used to identify the filter's poles and zeros. Specifically, the roots (values that make the expression zero) of the numerator identify zeros, and roots of the denominator identify poles. Zeros constitute feedforward paths and poles constitute feedback paths. By tracing the contour along the unit circle, the frequency response of the filter can be determined.

A filter is canonical if it contains the minimum number of delay elements needed to achieve its output. If the values of the coefficients are changed, the filter's response is altered. A filter is stable if its impulse response approaches zero as n goes to infinity. Convolution provides the means for implementing a filter directly from the impulse response; convolving the input signal with the filter impulse response gives the filtered output. In other words, convolution acts as the difference equation, and the impulse response acts in place of the difference equation coefficients in representing the filter. The choice of using a difference equation or convolution in designing a filter depends on the filter's architecture, as well as the application.

FIR Filters

As noted, the general difference equation can be written:

$$y(n) + b_1 y(n-1) + b_2 y(n-2) + \ldots + b_N y(n-N) =$$
$$a_0 x(n) + a_1 x(n-1) + a_2 x(n-2) + \ldots + a_M x(n-M).$$

Consider the general difference equation without b_i terms:

$$y(n) = \sum_{i=0}^{M} a_i x(n-i)$$

and its transfer function in the z domain:

$$H(z) = \sum_{i=0}^{M} a_i z^{-i}.$$

There are no poles in this equation, hence no feedback elements. The result is a nonrecursive filter. Such a filter would take the form:

$$y(n) = ax(n) + bx(n-1) + cx(n-2) + dx(n-3)\ldots$$

Any filter operating on a finite number of samples is known as a finite impulse response (FIR) filter.

As the name FIR implies, the impulse response has finite duration. Furthermore, an FIR filter can have only zeros outside the origin, it can have a linear phase, it responds to an impulse once, and it is always stable. Because it does not use feedback, it is called a nonrecursive filter. A nonrecursive structure is always an FIR; however, an FIR does not always use a nonrecursive structure.

Consider this introduction to the workings of FIR filters: we know that large differences between samples are indicative of high frequencies and small differences are indicative of low frequencies. A filter changes the differences between consecutive samples. The digital filter described by $y(n) = 0.5[x(n) + x(n-1)]$ makes the current output equal to half the current input plus half the previous input. Suppose this sequence is input: 1, 8, 6, 4, 1, 5, 3, 7; the difference between consecutive samples ranges from 2 to 7. The first two numbers enter the filter and are added and multiplied: $(1 + 8)(0.5) = 4.5$. The next computation is $(8 + 6)(0.5) = 7.0$. After the entire sequence has passed through the filter the sequence is: 4.5, 7, 5, 2.5, 3, 4, and 5. The new inter-sample difference ranges from 0.5 to 2.5; this filter averages the current sample with the previous sample. This averaging smoothes the output signal, thus attenuating high frequencies. In other words, the circuit is a lowpass filter.

More rigorously, the filter's difference equation is:

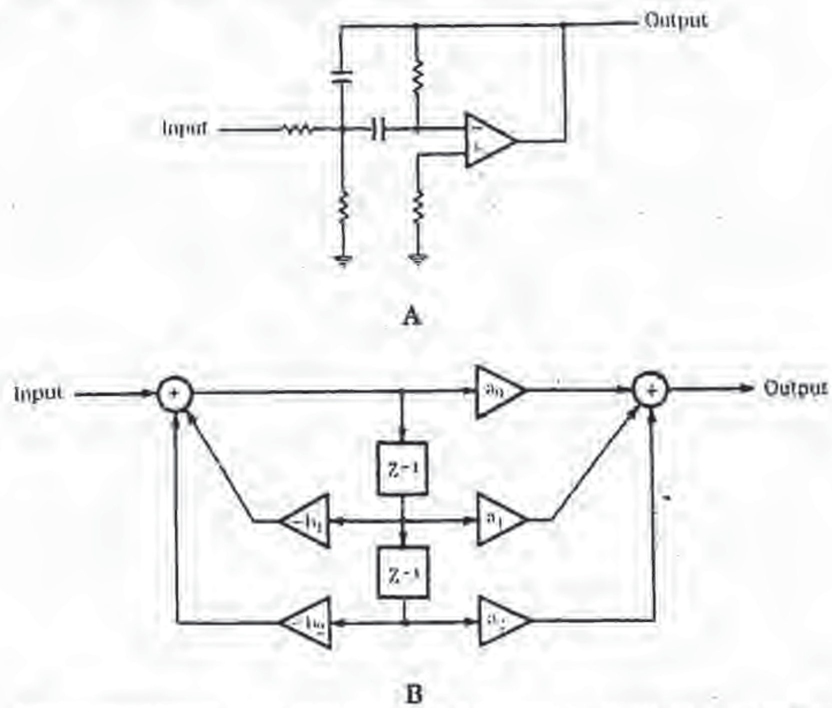
$$y(n) = 0.5[x(n) + x(n-1)].$$

Transformation to the z-domain yields:

$$Y(z) = 0.5[X(z) + z^{-1}X(z)].$$

The transfer function can be written:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{(1 + z^{-1})}{2} = \frac{(z+1)}{2z}.$$



15-14 A comparison of second-order analog and digital filters. A. A second-order analog filter. B. IIR biquadratic second-order filter section.

Filter Applications

An example of second-order analog filter is shown in Fig. 15-14A, and an IIR filter is shown in Fig. 15-14B; this is a bi-quadratic filter section. Coefficients determine the filter's response; in this example, with appropriate selection of the five multiplication coefficients, highpass, lowpass, bandpass, and shelving filters can be obtained. A digital audio processor might have several of these sections at its disposal. By providing a number of presets, users can easily select frequency response, bandwidth, and phase response of a filter. In this respect, a digital filter is more flexible than an analog filter that has relatively limited operating parameters. However, a digital filter requires considerable computation, particularly in the case of swept equalization. As the center frequency is moved, new coefficients must be calculated—not a trivial task. To avoid quantization effects (sometimes called zipper noise) filter coefficients and amplitude scaling coefficients must be updated at a theoretical rate equal to the sampling rate; in practice, an update rate equal to one-half or one-fourth the sampling rate is sufficient. To accomplish even this, coefficients are often obtained through linear interpolation; the range must be limited to ensure that filter poles do not momentarily pass outside the unit circle, causing transient instability.

Adaptive filters automatically adjust their parameters according to optimization criteria. They do not have fixed coefficients; instead, values are calculated during operation. Adaptive filters thus consist of a filter section and a control unit used to calculate coefficients. Often, the algorithm used to compute coefficients attempts to minimize the difference between the output signal and a reference signal. In general, any filter type can be used, but in practice, adaptive filters often use a transversal structure as well as lattice and ladder structures. Adaptive filters are used for applications such as echo and noise cancelers, adaptive line equalizers, and prediction.

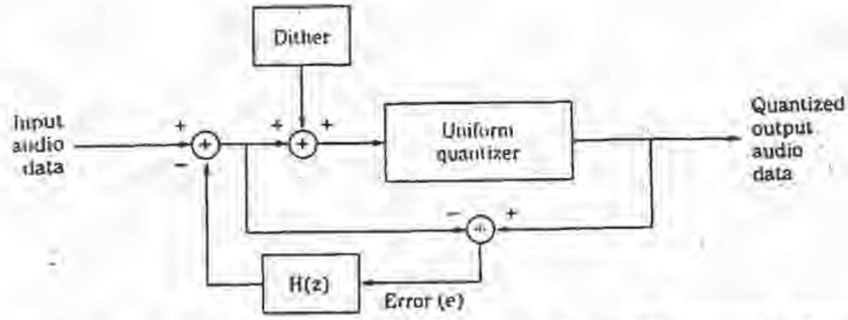
A transversal filter is a FIR filter in which the output value depends on both the input value, and a number of previous input values held in memory. Inputs are multiplied by coefficients and summed by an adder at the output. Only the input values are stored in delay elements; there are no feedback networks used, hence it is an example of a nonrecursive filter. As described in chapter 4, this architecture is used extensively to implement lowpass filtering with oversampling.

In practice, digital oversampling filters often use a cascade of FIR filters, designed so the sampling rate of each filter is a power of two higher than the previous filter. The number of delay blocks (tap length) in the FIR filter determines the passband flatness, transition band slope and stopband rejection; there are $M + 1$ taps in a filter with M delay blocks. Most digital filters are dedicated chips; however, general purpose DSP chips can be used to run custom filter programs.

The block diagram of a dedicated digital filter (oversampling) chip is shown in Fig. 15-15. It demonstrates the practical implementation of DSP techniques. A central processor performs computation while peripheral circuits accomplish input/output and other functions. The filter's characteristic is determined by the coefficients stored in ROM; the multiplier/accumulator performs the essential arithmetic operations; the shifter manages data during multiplication; the RAM stores intermediate computation results; a microprogram stored in ROM controls the filter's operation. The coefficient word length determines filter accuracy, and stopband attenuation. A filter can have, for example, 293 taps and a 22-bit coefficient; this would yield a passband flat to within ±0.00001 dB, with stopband suppression greater than 120 dB. Word length of the audio data increases during multiplication (length is the sum of the input words); truncation would result in quantization error thus the data must be rounded or dithered. Noise shaping can be applied at the accumulator, using an IIR filter to redistribute the noise power, primarily placing it outside the audio band. Noise shaping is discussed in chapter 16.

Sources of Errors

The DSP computation required to process an audio signal can result in noise and distortion unless precautions are taken. In general, errors in digital processors can be classified as coefficient errors, limit cycle errors, overflow, truncation and round-off errors. Coefficient errors occur when a coefficient is not specified with sufficient accuracy; a resolution of 24 bits or more is required for computations on 16-bit audio samples. Limit cycle error might occur when a signal is removed from a filter, leaving a decaying sum. This decay might become zero or might oscillate at a constant amplitude, known as limit cycle oscillation. This effect can be eliminated, for example, by offsetting the filter's output so that truncation always produces a zero output.



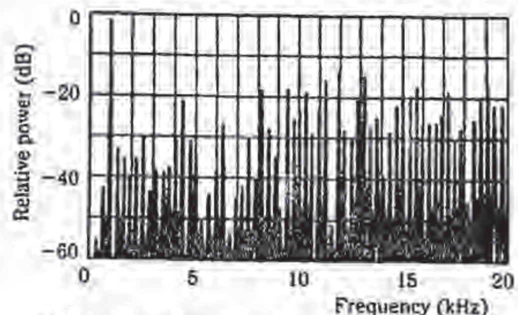
16-24 A requantization topology showing dithering and noise shaping. This processing reduces quantization distortion artifacts and can be used to reduce the noise floor in perceptually critical frequency regions.

but the higher frequency dither signal is shaped to even higher frequencies. However, correlation can result in higher overall noise. In this example, triangular pdf dither with a white spectrum appears to yield the best results.

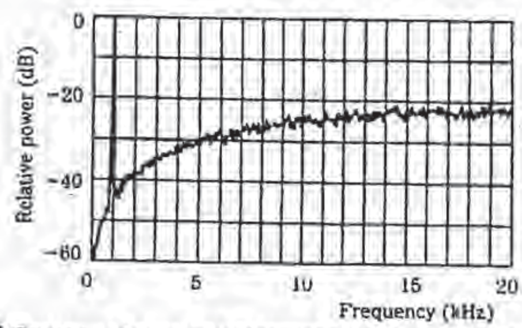
Psychoacoustically Optimized Noise Shaping

It is the goal of noise-shaping systems to dither the audio signal, then shape quantization noise to yield a less audible noise floor. These systems consider the fact that total noise power does not fully describe audibility of noise; perceived loudness also depends on spectral characteristics. Oversampling noise shapers reduce audio-band quantization noise and increase noise beyond the audio band, where it is inaudible. Nonoversampling noise shapers only redistribute noise energy within the audio band itself. For example, the difference in quantization noise between a 20-bit input signal and 16-bit output signal can be reshaped to minimize its audibility. In particular, psychoacoustically optimized noise-shaping systems use a feedback filter designed to shape the noise according to an equal loudness contour or other perceptual weighting function. In addition, such systems can use masking properties to conceal requantization noise.

Sixteen-bit master recordings are not adequate for subsequent replication on 16-bit CDs. For example, when using a digital console or hard-disk workstation to add equalization, change levels, or perform other digital signal processing, error accumulates in the 16th bit due to computation. It is desirable to use a longer word length, such as 20 bits, that allows processing prior to 16-bit storage. Furthermore, with proper transfer, much information contained in the four LSBs can be conveyed in the upper 16 bits. However, the problem of transferring 20 bits to 16 bits is not trivial. Simple truncation of the four least-significant bits greatly increases distortion. If the 16th bit is rounded, the improvement is only modest. It is thus important to redither the signal during the requantization that occurs in the transfer; this provides the same benefits as dithering during the original recording. If the most significant bit has not been exercised in the recording, it is possible to bit-shift the entire



A Spectrum of signal with undithered noise shaper.



B Spectrum of signal with triangular pdf-dithered noise shaper.

16-25 Dither profoundly affects the spectrum of the signal output from a noise-shaping circuit.

program upward, thus preserving more of the dynamic range. This is accomplished with a simple gain change in the digital domain. It can be argued that in some cases, for example, when transferring from an analog master tape, a 20-bit interface and noise shaping are not needed because the tape's noise floor makes it self-dithering. However, even then it is important to preserve the analog noise floor, which contains useful audio information.

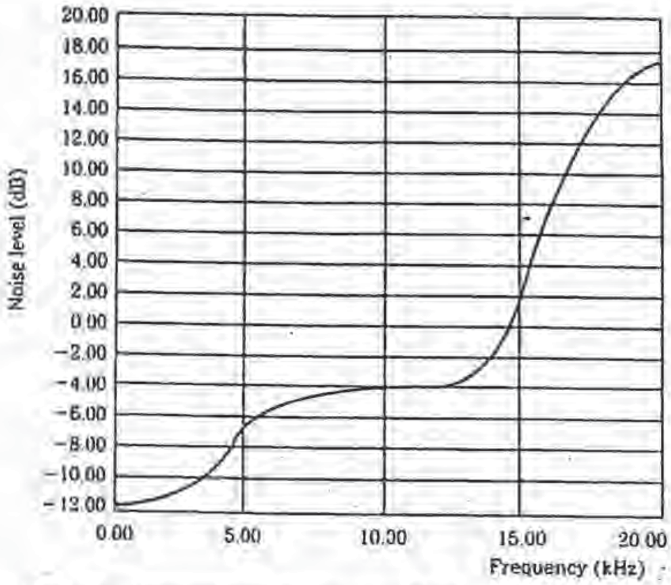
Nonoversampling noise-shaping systems are often used when converting a professional master recording to a consumer format such as CD. With linear conversion, and dither, a 16-bit recording can provide a distortion floor below −110 dB. Noise shaping cannot decrease total unweighted noise, but given a 20-bit master tape, subjective performance can be improved by decreasing noise in the critical 1- to 5-kHz region, at the expense of increasing noise in the noncritical 15-kHz region, and increasing total unweighted noise power as well. Because noise shaping removes requantization noise in the most critical region, this noise cannot mask audible details, thus improving subjective resolution. However, the benefit is realized only when output D/A converters exhibit sufficient low-level linearity, and high S/N ratio is available. Indeed, any subsequent requantization must preserve the most critical noise floor improvements, and not introduce other noise that would negate the advantage of a shaped noise floor. For example, 19-bit resolution in D/A converters can be required to fully preserve noise-shaping improvements in a 16-bit recording.

When reducing word length, the audio signal must be redithered for a level appropriate for the receiving medium, for example, 16 bits for CD storage; white triangular pdf dither can be used. A nonoversampling noise-shaping loop redistributes the spectrum of the requantization noise. As noted earlier in this chapter, sigma-delta noise shapers used in highly oversampled converters yield a contour with a gradually increasing spectral characteristic. This characteristic will not specifically reduce noise in the 1- to 5-kHz region. To take advantage of psychoacoustics, higher-order shapers are used in nonoversampling shapers to form more complicated weighting functions. In this way, the perceptually weighted output noise power is minimized. A digital filter $H(z)$ in a feedback loop (see Fig. 16-24) accomplishes this, in which the filter coefficients determine a response so that the output noise is weighted by $1 - H(z)$, the inverse of the desired psychoacoustic weighting function. The resulting weighted spectrum ideally produces a noise floor that is equally audible at all frequencies.

As Robert Wannamaker suggests, a suitable filter design begins with selection of a weighting function. This design curve is inverted, and normalized to yield a zero average spectral power density that represents the squared magnitude of the frequency response of the minimum-phase noise shaper. The desired response is specified, and an inverse Fourier transform is applied to produce an impulse response. The response is windowed to produce a number of filter coefficients corresponding to $1 - H(z)$; $H(z)$ is derived from this, yielding a FIR filter.

Theory shows that as very high-order filters $H(z)$ are used to approximate the optimal filter weighting function, the unweighted noise power increases, tending toward infinity with an infinite filter order. For example, although an optimal approximation might yield a 27-dB decrease in audible weighted noise (using an F-weighting curve that reflects the ear's high frequency roll off), other weighting functions must be devised, with more modest performance. For example, using a nine-coefficient FIR shaping filter, perceived noise can be decreased by 17 dB compared to unshaped requantization noise; total unweighted noise power is increased a reasonable 18 dB compared to an unshaped spectrum. In other words, the output is subjectively as quiet as an unshaped truncated signal with an additional three bits; in this way, 19-bit audio data can be successfully transferred to a 16-bit CD.

The balance of decrease in audible noise versus increase in total noise (at higher inaudible frequencies) is delicate. For example, a very high total noise power might register on digital audio meters or damage tweeters, and some listeners suggest that aggressively boosted high-frequency noise produces artifacts, or perhaps masks otherwise audible information. In practice, depending on the design, the weighting function often approximates a proprietary contour. For example, Fig. 16-26 shows a proprietary noise-shaping contour, plotted with linear frequency for clarity. In some cases, this curve is fixed; in other cases, the curve is adaptively varied according to signal conditions. Similarly, in some designs, an adaptive dither signal is correlated

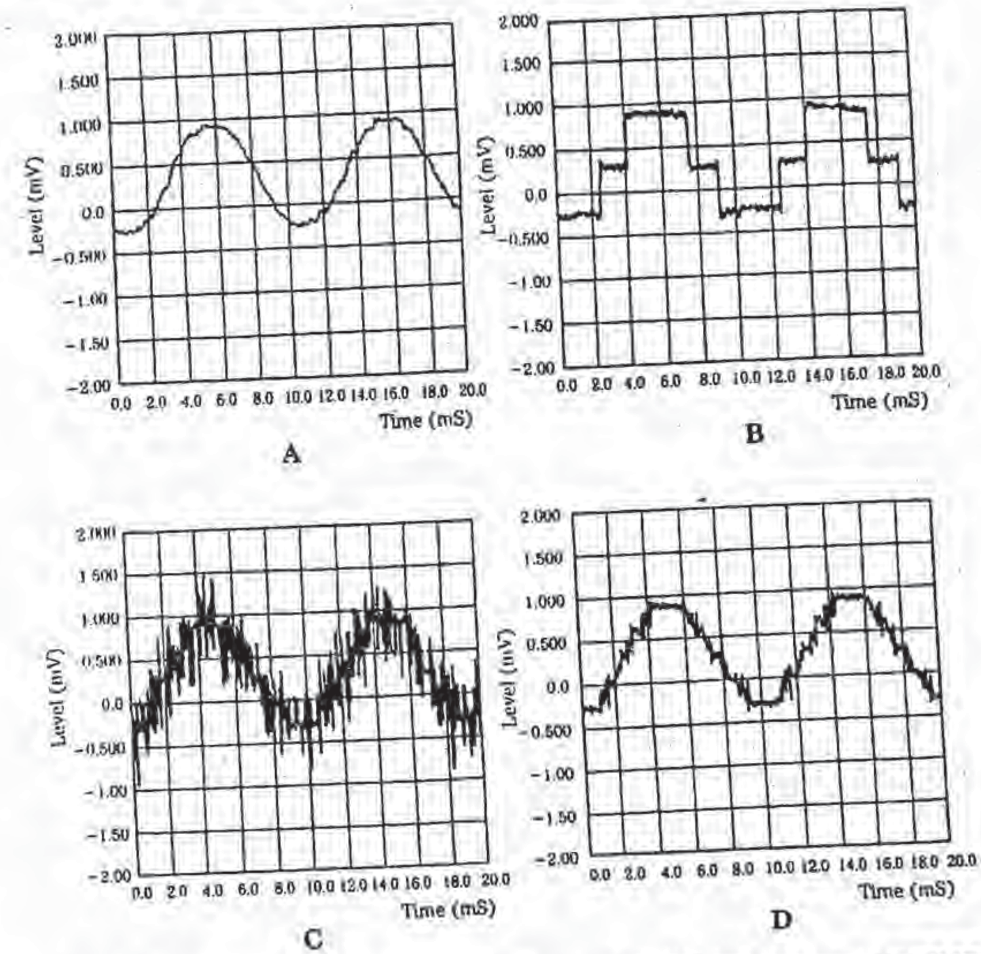


16-26 An equal-loudness noise-shaping curve. This frequency response plot uses a linear scale to better illustrate the high-frequency contour.

to the audio signal so the audio signal masks the added dither noise. For example, the audio signal can be spectrally analyzed so that dither frequencies slightly higher in frequency can be generated.

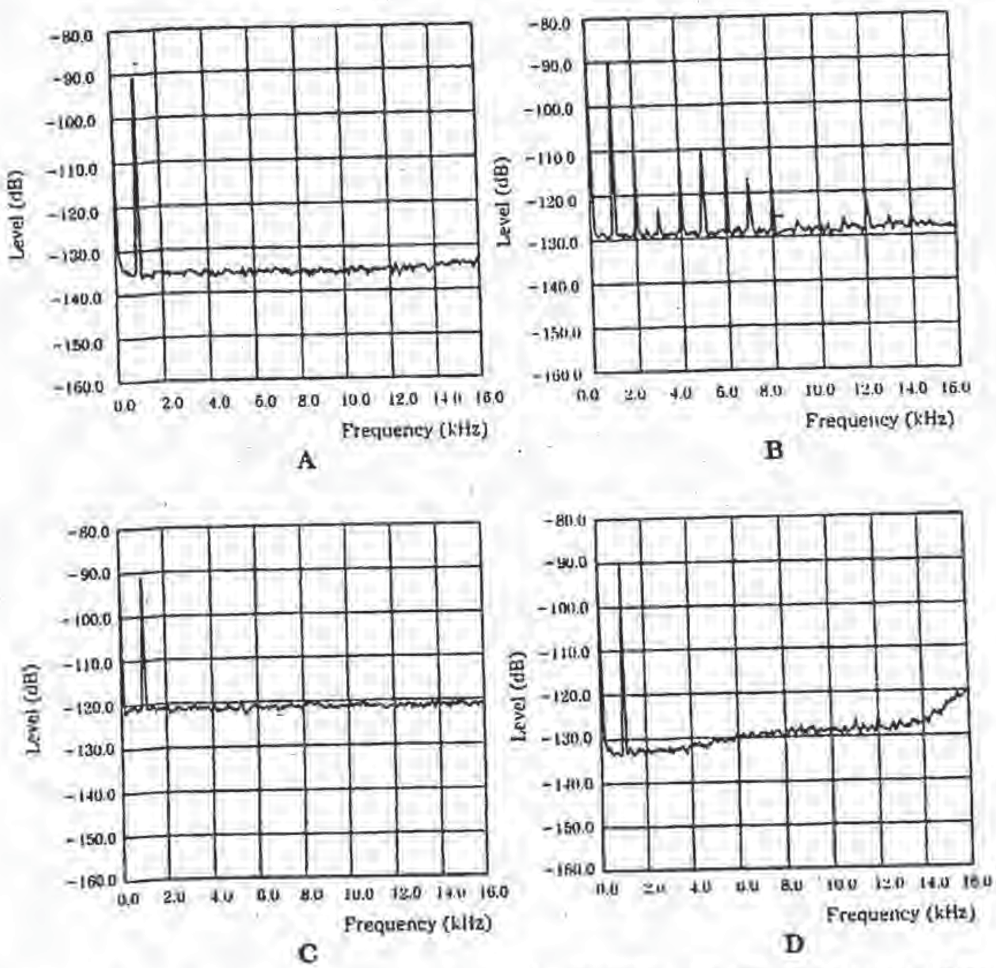
Figure 16-27 shows a 1-kHz sinewave with −90-dB amplitude; measurements are made with a 16-kHz lowpass filter, to approximate the ear's averaging response. A 20-bit recording is quite accurate; when truncated to 16-bits, quantization is clearly evident; when dithered (±1 LSB triangular pdf) to 16-bits, quantization noise is alleviated, but noise is increased; when noise shaping is applied, the noise in this lowpass filtered measurement is reduced. This 16-bit representation is quite similar to the original 20-bit representation. Figure 16-28 shows the spectrum of the same −90-dB sinewave, with the four representations. The 20-bit recording has low error and noise; truncation creates severe quantization error; dithering removes the error but increases noise; noise shaping reduces low- and mid-frequency noise, with an increase at higher frequencies.

In one implementation of a psychoacoustic noise shaper, adaptive error-feedback filters are used to optimize the requantization noise spectrum according to equal loudness contours as well as masking analysis of the input signal. An algorithm analyzes the signal's masking properties to calculate simultaneous masking curves. These are adaptively combined with equal loudness curves to calculate the noise-shaping filter's coefficients, to yield the desired contour. This balance is dynamically and continuously varied according to the power of the input signal; for example,



16-27 An example of noise shaping showing a 1-kHz sinewave with −90-B amplitude; measurements are made with a 16-kHz lowpass filter. A. Original 20-bit recording. B. Truncated 16-bit signal. C. Dithered 16-bit signal. D. Noise shaping preserves information in the lower 4 bits. Sony Corporation

when power is low, masking is minimal, so the equal loudness contour is used. Conversely, when power is high, masking is prevalent so the masking contour is more prominently used. The input signal is converted into critical bands, convolved with critical band masking curves, and converted to linear frequency to form the masking contour and hence the noise-shaping contour. In other words, masking analysis follows the same processing steps as used in perceptual coding.



16-28 An example of noise shaping showing the spectrum of a 1 kHz, −90-dB sinewave (from Fig. 16-27). A. Original 20-bit recording. B. Truncated 16-bit signal. C. Dithered 16-bit signal. D. Noise shaping reduces low and mid frequency noise, with an increase at higher frequencies. Sony Corporation

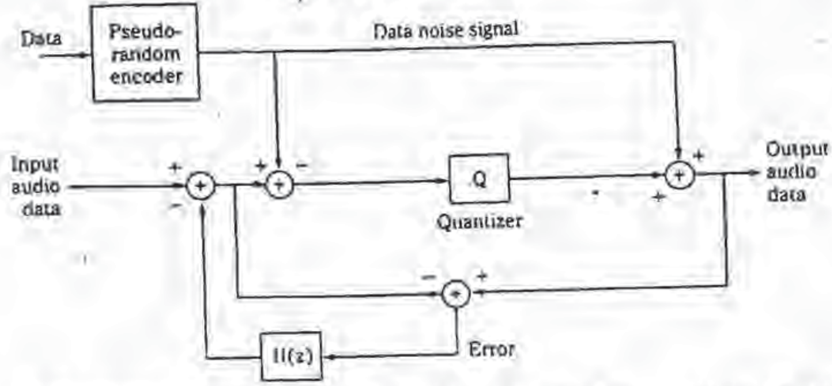
Buried Data Technique

With proper dithering and noise shaping, dynamic range can be improved. However, processing can also be applied to use this dynamic range for purposes other than conventional audio headroom. Michael Gerzon and Peter Craven have demonstrated how variable-rate data can be "buried" in a data stream. The data is coded with psychoacoustic considerations so the data is inaudible under the masking curve of the audio program; the added data signal is randomized to appear like shaped noise. For example, the method could be used to place new information on conventional audio CDs, without significantly degrading the quality of the audio program. In particular, the coding technique replaces several of the least-significant bits of the 16-bit format with independent data. Clearly, if unrelated data simply displaced audio data, and the disc was played in a conventional CD player, the result would be unlistenable. For example, nonstandard data in the four least-significant bits would add about 27 dB of noise to the music, as well as distortion caused by truncating the 16-bit audio signal. The buried data method makes buried data discs compatible with conventional CD players.

The buried data is first coded to be pseudo-random, to make it noise-like. This signal is used as subtractive dither to remove the artifacts caused by quantization; specifically, the data dither is subtracted prior to quantization, then added after quantization, replacing the several least-significant bits of the output signal. In addition, noise shaping is applied in a loop around the quantizer to lower the perceived noise, as shown in Fig. 16-29. As a result, the noise created by four bits of buried data per channel (conveying 352.8 kbps with stereo channels) is reduced to yield an overall S/N ratio of about 91 dB, a level that is similar to conventional CDs. Two bits of buried data provides a buried channel rate of 176.4 kbps, while maintaining a S/N ratio of 103 dB. The method could variably "steal" bits from the original program only when their absence will be psychoacoustically masked by the music signal. The noise-shaping characteristic is varied according to the analyzed masking properties of the signal. The overall buried data rate could exceed 500 kbps, with 800 kbps possible during loud passages, depending on the music program. Combining methods, for example, buried data might consist of two 2-bit fixed channels, and a variable rate channel; side information would indicate the variable data rate. A buried data CD could be played in a regular CD player; the fidelity of music with limited dynamic range might not be affected at all.

More significantly, a CD player with appropriate decoding (or a player outputting buried data to an external decoder) could play the original music signal, and process buried data as well. The possibilities for buried data are numerous; many audio improvements can be more useful than the lost dynamic range. For example, buried 4-bit data could be used to convey multiple (5.1 channel) audio channels for surround sound playback; the main left/rights channels are conventionally coded, the buried data carries four additional channels. A 5.1 disc would compatibly deliver stereo reproduction with a conventional CD player, and surround sound with a 5.1 CD player. Alternatively, one or two bits of buried data could carry dynamic range compression or expansion information. Depending on the playback circumstances, the dynamic range of the music could be adjusted for the most desirable characteristics. Because the range algorithms are calculated prior to playback, they are much more effective than conventional real-time dynamic processing. Buried data could convey additional high-frequency information above the Nyquist frequency, and provide a gentle bandlimiting roll-off rate. Any of these applications could be combined, within the limits of the buried data's rate. For example, two ambience channels and dynamic range control data could be delivered simultaneously.

Spread spectrum — 4 LSBs ?



16-29 A buried channel encoder converts added data to a pseudo-random noise signal, which is used as a dither signal. This is subtracted from the audio signal prior to quantization and added to the signal after quantization. Noise shaping is performed around the quantizer. Gerzon and Craven

Conclusion

In addition to obsoleting brick-wall analog filters, low-bit A/D converters surpass conventional multibit A/D converters by achieving increased resolution. Specifically, in-band noise can be made quite small. This benefit is provided by SDM; the same circuit that codes the signal into a low-bit stream also shifts the out-of-band noise components. Similarly, highly oversampling D/A converters using noise shaping and low-bit conversion largely surpass the performance of multibit D/A converters. In phase linearity, amplitude linearity, noise, long-term stability, and other parameters, A/D and D/A converters using low-bit architectures offer significant advantages. Noise shaping is also critical when reducing word length during data transfer; with nonoversampling noise shaping and dither, 19 bits of perceived resolution can be coded in a 16-bit storage medium. These applications all underscore the power of digital signal processing.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

ACKNOWLEDGMENTS

The list of people who had a hand in this book seems unending, but all are worthy of mention. I would like to thank Don Alvarez, Ross Anderson, Karl Barrus, Steve Bellovin, Dan Bernstein, Eli Biham, Joan Boyar, Karen Cooper, Whitfield Diffie, Joan Feigenbaum, Phil Karn, Neal Koblitz, Xuija Lai, Tom Leranth, Mark Markowitz, Ralph Merkle, Bill Patton, Peter Pearson, Mark Riordan, and Marc Schwartz for reading and editing all or parts of the manuscript; Lawrie Brown, Leisa Condie, Peter Gutmann, Alan Insley, Xuija Lai, Peter Pearson, Ken Pizzini, Richard Outerbridge, RSA Data Security Inc., Michael Wood, and Phil Zimmermann for providing source code; the readers of sci.crypt for commenting on ideas and answering questions, Paul MacNerland for creating the figures; Randy Seuss for providing Internet access; Jeff Duntemann and Jon Erickson for helping me find a publisher; Paul Farrell for editing this book; assorted random Insleys for the impetus, encouragement, support, conversations, friendship, and dinners; and AT&T Bell Labs for firing me and making this all possible. These people helped to create a far better book than I could have done alone.

Bruce Schneier
Oak Park, Ill.

Applied Cryptography

CHAPTER 1

Foundations

1.1 TERMINOLOGY

Sender and Receiver

Suppose someone, whom we shall call the **sender**, wants to send a message to someone else, whom we shall call the **receiver**. Moreover, the sender wants to make sure an intermediary cannot affect the message in any way: specifically, the receiver cannot intercept and read the message, intercept and modify the message, or fabricate a realistic-looking substitute message.

Messages and Encryption

A message is called either **plaintext** or **cleartext**. The process of disguising a message in such a way as to hide its substance is called **encryption**. An encrypted message is called **ciphertext**. The process of turning ciphertext back into plaintext is called **decryption**. This is shown in Figure 1.1.

The art and science of keeping messages secure is called **cryptography**, and it is practiced by **cryptographers**. **Cryptanalysts** are practitioners of **cryptanalysis**, the art and science of breaking ciphertext, i.e., seeing through the disguise. The branch of mathematics embodying both cryptography and cryptanalysis is called **cryptology**, and its practitioners are called **cryptologists**. These days almost all cryptologists are also theoretical mathematicians—they have to be.

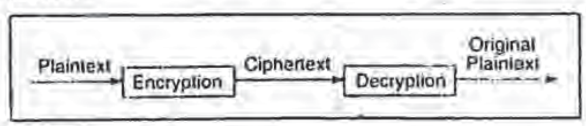


Figure 1.1

BEST AVAILABLE COPY

EB9 ~~Doc~~

Plaintext is denoted by P. It can be a stream of bits, a text file, a stream of digitized voice, or a digital video image. As far as a computer is concerned, P is simply binary data. (Outside the historical chapter, this book concerns itself with binary data.) The plaintext can be intended for either transmission or storage. In any case, P is the message to be encrypted.

Ciphertext is denoted by C. It is also binary data, sometimes the same size as P, sometimes larger. (By combining compression and encryption, C may be smaller than P. However, encryption alone usually does not accomplish this.) The encryption function E operates on P to produce C. Or, in mathematical notation:

$$E(P) = C$$

In the reverse process, the decryption function D operates on C to produce P:

$$D(C) = P$$

Since the whole point of encrypting and then decrypting a message is to recover the original plaintext, the following identity must hold true:

$$D(E(P)) = P$$

Algorithms and Ciphers

A **cryptographic algorithm,** also called a **cipher,** is the mathematical function used for encryption and decryption. To encrypt a plaintext message, apply an **encryption algorithm** to the plaintext. To decrypt a ciphertext message, apply a **decryption algorithm** to the ciphertext.

If the security of an algorithm is based on keeping the nature of the algorithm secret, it is called **restricted.** Restricted algorithms have historical interest, but by today's data security standards they provide woefully inadequate security. A large or changing group of users cannot use them, because users will eventually reveal the secret. When they do, the whole security of the system fails. More important, most restricted cryptosystems are trivial to break by experienced cryptanalysts. Despite this, restricted algorithms are enormously popular for low-security applications. Zenith's video-scrambling technique is an example of a restricted algorithm.

For real security, all modern encryption algorithms use a **key,** denoted by k. This key can take on one of many values (a large number is best). The range of possible values of the key is called the **keyspace.**

The value of the key affects the encryption and decryption functions, so the encryption and decryption functions now become:

$$E_k(P) = C$$
$$D_k(C) = P$$

And if the encryption key and the decryption key are the same, then:

$$D_k(E_k(P)) = P$$

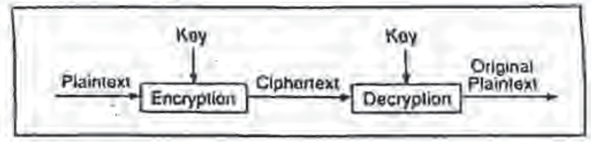


Figure 1.2
Encryption and decryption with key.

This is shown in Figure 1.2.

There are algorithms where the encryption key and the decryption key come in pairs (see Figure 1.3). That is, the encryption key, k_1, is different from the corresponding decryption key, k_2. In this case:

$$E_{k_1}(P) = C$$
$$D_{k_2}(C) = P$$
$$D_{k_2}(E_{k_1}(P)) = P$$

Symmetric Algorithms and Public-Key Algorithms

There are two general forms of key-based algorithms: symmetric and public-key. **Symmetric algorithms** are algorithms where the encryption key can be calculated from the decryption key and vice versa. In many such systems, the encryption key and the decryption key are the same. These algorithms, also called **secret-key algorithms, single-key algorithms,** or **one-key algorithms,** require the sender and receiver to agree on a key before they pass messages back and forth. This key must be kept secret. The security of a symmetric algorithm rests in the key; divulging the key means that anybody could encrypt and decrypt messages in this cryptosystem.

Encryption and decryption with a symmetric algorithm are denoted by:

$$E_k(P) = C$$
$$D_k(C) = P$$

Symmetric algorithms can be divided into two categories. Some operate on the plaintext a single bit at a time; these are called **stream algorithms** or **stream ciphers.** Others operate on the plaintext in groups of bits. The groups of bits are called **blocks,** and the algorithms are called **block algorithms** or **block ciphers.** For algorithms that are implemented on computer, a typical block size is 64 bits—large enough to preclude analysis and small enough to be workable. In both block and stream algorithms, the same key is used for both encryption and

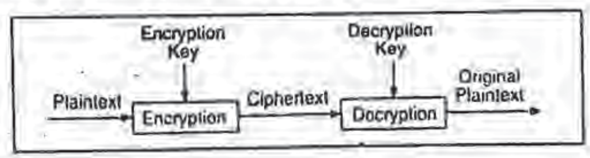


Figure 1.3
Encryption and decryption with two keys.

when cryptanalysts have a tamperproof box that does automatic decryption, the job is to deduce the key.

Given: $C_1, P_1 = D_k(C_1), C_2, P_2 = D_k(C_2), \ldots C_i, P_i = D_k(C_i)$
Deduce: k

This attack is primarily applicable to public-key cryptosystems and will be discussed in Section 12.4. A chosen-ciphertext attack also works against symmetric algorithm, but due to the symmetry of these cryptosystems it is equivalent in complexity to a chosen-plaintext attack.

6. **Chosen-key attack.** This is not an attack when you're given the key. It's strange and obscure, not very practical, and is discussed in Section 10.1.

Known-plaintext attacks and chosen-plaintext attacks are more common than you might think. It is not unheard of for a cryptanalyst to get a plaintext message that has been encrypted or to bribe someone to encrypt a chosen message. You may not even have to bribe someone; if you give a message to an ambassador, you will probably find that it gets encrypted and sent back to the United States for consideration. Many messages have standard beginnings and endings that might be known to the cryptanalyst. Encrypted source code is especially vulnerable because of the regular appearance of keywords: `#define`, `struct`, `else`, `return`. Encrypted executable code has the same kinds of problems, called protocols, loop structures, etc. David Kahn's books [462,463,464] have some historical examples of these kinds of attacks.

One of the fundamental axioms of cryptography is that the enemy is in full possession of the details of the algorithm and lacks only the specific key used in the encryption. (Of course, one would assume that the CIA does not make a habit of telling Mossad about its cryptographic algorithms, but Mossad probably finds out anyway.) While this is not always true in real-world cryptanalysis, it is always true in academic cryptanalysis; and it's a good assumption to make in real-world cryptanalysis. If others can't break an algorithm even with knowledge of how it works, then they certainly won't be able to break it without that knowledge.

Cryptanalysts don't always have access to the algorithm—for example, when the United States broke the Japanese diplomatic code, PURPLE, during World War II [462]—but they often do. If the algorithm is being used in a commercial security program, it is simply a matter of time and money to disassemble the program and recover the algorithm. If the algorithm is being used in a military communications system, it is simply a matter of time and money to buy (or steal) the equipment and reverse-engineer the algorithm. There have been many historical instances when cryptanalysts did not know the encryption algorithms; sometimes they broke the algorithm anyway, and sometimes they did not. In any case, it is unrealistic to rely on it.

Those who claim to have an unbreakable cipher simply because they can't break it are either geniuses or fools. Unfortunately, there are more of the latter in the world. Beware of people who extol the virtues of their algorithms, but refuse to make them public; trusting their algorithms is like trusting snake oil.

On the other hand, a good algorithm can be made public without worry. You can send it to your adversaries, publish it in a magazine, or shout it from the rooftops. It doesn't matter; even the designer of the algorithm can't decrypt messages without the key.

Good cryptographers rely on peer review to separate the good algorithms from the bad.

Security of Cryptosystems

Different cryptosystems have different levels of security, depending on how hard they are to break. As we will see, all algorithms but one are theoretically **breakable**, given enough time and computing resources. If the time and money required to break an algorithm is more than the value of the encrypted data, then it is probably safe. Computers are becoming increasingly faster and cheaper. At the same time, the value of the data decreases over time. It is important that those two lines never cross.

Some algorithms are only breakable with the benefit of more time than the universe has been in existence and a computer larger than all the matter in the universe. These algorithms are theoretically breakable, but not breakable in practice. An algorithm that is not breakable in practice is **secure**.

An algorithm is **unconditionally secure** if, no matter how much ciphertext a cryptanalyst has, there is not enough information to recover the plaintext. In point of fact, only a one-time pad (see Section 1.2.4) is unbreakable given infinite resources. Cryptography is more concerned with cryptosystems that are computationally unfeasible to break. An algorithm is considered **computationally secure**, or **strong**, if it cannot be broken with available (current or future) resources. Exactly what constitutes "available resources" is open to interpretation.

The amount of computing time and power required to recover the encryption key is called the **work factor**, and is expressed as an order of magnitude. If an algorithm has a work factor of 2^{128}, then 2^{128} operations are required to break the algorithm. These operations can be very complex and time-consuming, but details of the operations are left to the implementation. Still, if you assume that you have enough computing speed to perform a million of them every second, and you set a million parallel processors against the task, it will still take over 10^{19} years to recover the key. (For comparison's sake, the age of the universe is estimated at 10^{10} years.) I would consider an algorithm that takes a billion times the age of the universe to break to be computationally secure.

While the work factor required to break a given algorithm is constant (that is, until some cryptanalyst finds a better cryptanalytic attack), computing power is anything but. During the last half-century we have seen phenomenal advances in computing power, and there is no reason to think that it will change anytime soon. Many cryptanalytic attacks are perfect for parallel machines: the task can be broken down into billions of tiny pieces and none of the processors needs to interact with another. Pronouncing that an algorithm is secure simply because it is unfeasible to break, given current technology, is dicey at best. Good cryptosystems are designed to be unfeasible to break with the computing power that is expected to evolve many years in the future.

decryption. (Before computers, algorithms generally operated on plaintext one character at a time. You can either think of this as a stream algorithm operating on a stream of characters or as a block algorithm operating on 8-bit blocks.)

Public-key algorithms are different. They are designed so that the key used for encryption is different from the key used for decryption. Furthermore, the decryption key cannot be (at least in any reasonable amount of time) calculated from the encryption key. They are called public-key systems because the encryption key can be made public: a complete stranger can use the encryption key to encrypt a message, but only someone with the corresponding decryption key can decrypt the message. In these systems, the **encryption key** is often called the **public key**, and the **decryption key** is often called the **private key**. In instances when both keys must be kept secret, sometimes the terms "encryption key" and "decryption key" will be used. The private key is sometimes also called the secret key, but to avoid confusion with symmetric algorithms, that moniker won't be used here.

Encryption using public key k is denoted by:

$$E_k(P) = C$$

Even though the public key and private key are different, decryption with the corresponding private key is denoted by:

$$D_k(C) = P$$

Sometimes, messages will be encrypted with the private key and decrypted with the public key; this is used in digital signatures (see Section 2.6). Despite the possible confusion, these operations will be denoted by, respectively:

$$E_k(P) = C$$
$$D_k(C) = P$$

In this book, "algorithm" will refer specifically to the mathematical transformations for encryption and decryption. "Cryptosystem" will refer to the algorithm, plus the way in which it is implemented. "Cipher" will often be used to refer to a family of algorithms, e.g., "block cipher."

Cryptanalysis

The primary purpose of cryptography is to keep the plaintext (or the key, or both) secret from **eavesdroppers** (also called **adversaries**, **attackers**, **interceptors**, **interlopers**, **intruders**, **opponents**, or simply **the enemy**). Cryptanalysis is the science of recovering the plaintext of a message without the key. Successful cryptanalysis may recover the plaintext or the key. It also may find weaknesses in a cryptosystem that eventually lead to the above results.

An attempted cryptanalysis is called an **attack**. A successful attack is called a **method**. An attack assumes that the cryptanalyst has the details of the cryptographic algorithm. While this is not always the case in real-life

cryptanalysis, it is the conventional assumption for academic cryptanalysis. It is a good assumption to make; if your security depends on the secrecy of the algorithm, then there is only minimal security.

There are six general types of cryptanalytic attacks, listed in order of power. Each of them assumes that the cryptanalyst has complete knowledge of the encryption algorithm used:

1. **Ciphertext-only attack.** In this attack, the cryptanalyst has the ciphertext of several messages, all of which have been encrypted using the same encryption algorithm. The cryptanalyst's job is to recover the plaintext of as many messages as possible, or better yet deduce the key (or keys) used to encrypt the messages in order to decrypt other messages encrypted with the same keys.

   Given: $C_1 = E_k(P_1), C_2 = E_k(P_2), \ldots C_i = E_k(P_i)$
   Deduce: Either $P_1, P_2, \ldots P_i$; k; or an algorithm to infer P_{i+1} from $C_{i+1} = E_k(P_{i+1})$

2. **Known-plaintext attack.** The cryptanalyst not only has access to the ciphertext of several messages but also to the plaintext of those messages. The job is to deduce the key (or keys) used to encrypt the messages or an algorithm to decrypt any new messages encrypted with the same key.

   Given: $P_1, C_1 = E_k(P_1), P_2, C_2 = E_k(P_2), \ldots P_i, C_i = E_k(P_i)$
   Deduce: Either k, or an algorithm to infer P_{i+1} from $C_{i+1} = E_k(P_{i+1})$

3. **Chosen-plaintext attack.** Cryptanalysts not only have access to the ciphertext and associated plaintext for several messages, but they also choose the encrypted plaintext. This is more powerful than a known-plaintext attack, because cryptanalysts can choose specific plaintext blocks to encrypt, ones that might yield more information about the key. The job is to deduce the key (or keys) used to encrypt the messages or an algorithm to decrypt any new messages encrypted with the same key.

   Given: $P_1, C_1 = E_k(P_1), P_2, C_2 = E_k(P_2), \ldots P_i, C_i = E_k(P_i)$, where the cryptanalysts choose $P_1, P_2, \ldots P_i$
   Deduce: Either k, or an algorithm to infer P_{i+1} from $C_{i+1} = E_k(P_{i+1})$

4. **Adaptive-chosen-plaintext attack.** This is a special case of a chosen-plaintext attack. Not only can cryptanalysts choose the plaintext that is encrypted, but they can modify the choice based on the results of previous encryption. In a chosen-plaintext attack, cryptanalysts might just be able to choose one large block of plaintext to be encrypted; in an adaptive-chosen-plaintext attack they can choose a smaller block of plaintext and then choose another based on the results of the first, etc.

5. **Chosen-ciphertext attack.** Cryptanalysts can choose different ciphertexts to be decrypted and have access to the decrypted plaintext. In an instance

Historical Terms

There are other cryptographic terms. A cryptosystem is also called a code or a cipher. Encrypting is also called **encoding** or **enciphering**, and decrypting is also called **decoding** or **deciphering**.

Historically, a code refers to a cryptosystem that deals with linguistic units: words, phrases, sentences, etc. For example, the word "MULBERRY" might be the ciphertext for the entire phrase "TURN LEFT 90 DEGREES"; the word "LOLLIPOP" might be the ciphertext for "TURN RIGHT 90 DEGREES"; and the words "BENT EAR" might be the ciphertext for "HOWITZER." Codes of this type are not discussed in this book; they are discussed in [462,463].

The word "cipher" has historically been used to refer to cryptosystems in which individual letters are swapped and substituted or otherwise scrambled to hide the plaintext. This is what this book is about.

Ciphers were useful because they were general-purpose. If there was no entry in a codebook for "ANTEATERS," then you couldn't say it. On the other hand, any message can be encrypted with the cipher.

1.2 CLASSICAL CRYPTOGRAPHY

Before computers, cryptography consisted of character-based cryptosystems. Different cryptographic algorithms either substituted characters for one another or transposed characters with one another. The better cryptosystems did both—many times each.

Things are more complex these days, but the philosophy has remained the same. The primary change is that algorithms work on bits instead of characters. This is actually just a change in the alphabet size, from 26 elements to two elements and nothing more. Most good cryptographic algorithms still combine elements of substitution and transposition (there are exceptions).

1.2.1 Substitution Ciphers and Transposition Ciphers

Substitution Ciphers

A **substitution cipher** is one in which each character in the plaintext is substituted for another character in the ciphertext. This substitution serves to obscure the plaintext from everyone but the recipient, who inverts the substitution on the ciphertext to recover the plaintext.

In classical cryptography, there are four basic types of substitution ciphers:

- A **simple substitution cipher** is one in which a character of the plaintext is replaced with a corresponding character of ciphertext. The cryptograms in newspapers are simple substitution ciphers.
- A **homophonic substitution cipher** is like a simple substitution cryptosystem, except a single character of plaintext can map to one of several characters of ciphertext. For example, "A" could correspond to either 5, 13, 25, or 56, "B" could correspond to either 7, 19, 31, or 42, etc.

- A **polyalphabetic substitution cipher** is made up of multiple simple substitution ciphers. For example, there might be five different simple substitution ciphers used; the particular one used changes with the position of each character of the plaintext.
- A **polygram substitution cipher** is one in which blocks of characters are encrypted in groups. For example, "ABA" could correspond to "RTQ," "ABB" could correspond to "SLL," etc.

The famous **Caesar Cipher**, in which each plaintext character is replaced by the character three to the right mod 26 (A is replaced by D, B is replaced by E, . . . , W is replaced by Z, . . . , X is replaced by A, Y is replaced by B, and Z is replaced by C) is a simple substitution cipher.

ROT13 is a simple encryption program commonly found on UNIX systems. In this cipher, A is replaced by N, B is replaced by O, etc. Every letter is rotated thirteen places. This is a simple substitution cipher.

```
#include <stdio.h>
main()  /* streamlined version of copy input to output */
{
  int c;
  while ((c =getchar()) !=EOF)
  {
    if( c>= 'a' && c<= 'm' )
      c=c+13;
    else if( c >= 'n' && c <= 'z' )
      c=c-13;
    else if( c >= 'A' && c <= 'M' )
      c=c+13;
    else if( c >= 'N' && c <= 'Z' )
      c=c-13;
    putchar(c);
  }
}
```

Encrypting a file twice with ROT13 restores the original file.

$P = \mathrm{ROT13}(\mathrm{ROT13}(P))$

ROT13 is not intended for security; it is often used in electronic mail messages to hide potentially offensive text, to avoid giving away the solution to a puzzle, etc.

These ciphers can be easily broken because the cipher does not hide the underlying frequencies of the different letters of the plaintext. All it takes is about 25 English characters before a good cryptanalyst can reconstruct the plaintext [806]. A general algorithm for solving these sorts of ciphers can be found in [669].

Homophonic substitution ciphers were used as early as 1401 by the Duchy of Mantua [462]. They are much more complicated to break than simple substitution ciphers but still do not obscure all of the statistical properties of the plaintext language. With a known-plaintext attack, the ciphers are trivial to break. A ciphertext-only attack is harder, but only takes a few seconds on a computer. Details are in [710].

Polyalphabetic substitution ciphers were invented by Leon Battista in 1568 [462]. They were used by the Union army during the American Civil War. Despite the fact that they can be broken easily [475,355,360,462] (especially with the help of computers), many commercial computer security products use ciphers of this form [744]. (Details on how to break this encryption scheme as it was found in the WordPerfect word-processing program versions can be found in [80,84].) The Vigenère cipher and the Beaufort cipher are examples of polyalphabetic substitution ciphers.

Polyalphabetic substitution ciphers have multiple one-letter keys, each of which is used to encrypt one letter of the plaintext. The first key encrypts the first letter of the plaintext, the second key encrypts the second letter of the plaintext, and so on. After all the keys are used, the keys are recycled. If there were 20 one-letter keys, then every twentieth letter would be encrypted with the same key. This is called the **period** of the cipher. In classical cryptography, ciphers with longer periods were significantly harder to break than ciphers with short periods. With computers, there are techniques that can easily break substitution ciphers with very long periods.

A **running-key cipher**, in which one text is used to encrypt another text, is another example of this sort of cipher. Even though this cipher has a period the length of the text, it can also be broken easily [354,462].

Polygram substitution ciphers are ciphers in which groups of letters are encrypted together. The Playfair cipher, invented in 1854, was used by the British during World War I [462]. It encrypts pairs of letters together. Its cryptanalysis is discussed in [360,832,500]. The Hill cipher is another example of a polygram substitution cipher [436].

Transposition Ciphers

A **transposition cipher** is one in which the characters in the plaintext remain the same, but their order is shuffled around. In a simple columnar transposition cipher, the plaintext is written horizontally onto a piece of graph paper of fixed width, and the ciphertext is read off vertically (see Figure 1.4). Decryption is a matter of writing the ciphertext vertically onto a piece of graph paper of identical width and then reading the plaintext off horizontally. Cryptanalysis of these ciphers is discussed in [360,832].

The German ADFGVX cipher, used during World War I, is a transposition cipher (plus a simple substitution). It was a very complex algorithm for its day but was broken by Georges Painvin, a French cryptanalyst [462].

Although many modern cryptosystems use transposition, it is troublesome because it requires a lot of memory and sometimes requires messages to be a multiple of a certain length. Substitution is far more common.

Plaintext: COMPUTER GRAPHICS MAY BE SLOW BUT AT LEAST IT'S EXPENSIVE.

```
COMPUTERGR
APHICSMAYB
ESLOWBUTAT
LEASTITSEX
PENSIVE
```

Ciphertext: CAELP OPSEE MHLAN PIOSS UCWTI TSBIU EMUTE RATSG YAERB TX

Figure 1.4 Columnar transposition cipher

Rotor Machines

In the 1920s, various mechanical encryption devices were invented to automate the process of encryption. They were based on the concept of a **rotor**, a mechanical wheel that was wired to perform a general cryptographic substitution. For example, a rotor might be wired to substitute "F" for "A," "U" for "B," "L" for "C," etc. These devices used multiple rotors to implement a version of the Vigenère cipher with a very long period and were called **rotor machines**.

A rotor machine has a keyboard and a series of rotors. Each rotor has 26 positions and performs a simple substitution. The rotations of the rotors are a Caesar Cipher. It is the combination of all these rotors that makes the machine secure. Because the rotors all move, and at different rates, the period for an n-rotor machine is 26^n.

The best-known rotor device is the Enigma. The Enigma was used by the Germans during World War II. The basic idea was invented by Arthur Scherbius and Arvid Gerhard Damm in Europe. It was patented in the United States by Arthur Scherbius [772]. The Germans beefed up the basic design considerably for wartime use.

There was a plugboard that slightly permuted the plaintext. There was a reflecting rotor that forced each rotor to operate on each plaintext letter twice. As complicated as the Enigma was, it was broken during World War II. A team of Polish cryptographers broke a simplified Enigma; a British team, including Alan Turing, broke the actual Enigma. For explanations of how rotor ciphers work and how they were broken, see [462,45,301,258,500,740,873]. Two fascinating accounts of how the Enigma was broken are [438,464].

Further Reading

This is not a book about classical cryptography, so I will not dwell further on these subjects. Two excellent precomputer cryptology books are [360,832]. Dorothy Denning discusses many of these ciphers in [268], and [500] has some fairly complex mathematical analyses of the same ciphers. An article that presents a good overview of the subject is [356]. David Kahn's historical cryptography books are also excellent [462,463,464].

1.2.2 Computer Algorithms

There are many cryptographic algorithms. These are three of the most common:

- **DES** (Data Encryption Standard) is currently the most popular computer encryption algorithm. DES is a U.S.-government standard encryption algorithm and has been endorsed by the U.S. military for encrypting "Unclassified but Sensitive" information. It is a symmetric algorithm; the same key is used for encryption and decryption.
- **RSA** (named for its creators—Rivest, Shamir, and Adleman) is the most popular public-key algorithm. It can be used for both encryption and digital signatures.
- **DSA** (Digital Signature Algorithm, used as part of the Digital Signature Standard) is another public-key algorithm. It cannot be used for encryption, but only for digital signatures.

1.2.3 Simple XOR

It's an embarrassment to put this algorithm in a book like this because it's nothing more than a Vigenère cipher. It is included because it is so prevalent in commercial software packages, at least those in the MS-DOS and Macintosh worlds [847]. Unfortunately, if a software security program proclaims that it has a "proprietary" encryption algorithm—one that is significantly faster than DES—the odds are that it is some variant of this.

```
/* Usage:  crypto key input_file output_file  */

void main (int argc, char *argv[])
{
  FILE *fi, *fo;
  int *cp;
  int c;

  if (cp = argv[1])  {
    if ((fi = fopen(argv[2], "rb"))  != NULL)  {
      if ((fo = fopen(argv[3], "wb")) != NULL)  {
        while ((c = getc(fi)) != EOF)  {
          if (!*cp) cp = argv[1];
          c ^= *(cp++);
          putc(c,fo);
        }
        fclose(fo);
      }
      fclose(fi);
    }
  }
}
```


This is a symmetric algorithm; the same key is used for both encryption and decryption. The plaintext is being XORed with a keyword to generate the ciphertext. Since XORing the same value twice restores the original, encryption and decryption use exactly the same program:

P XOR $K = C$
C XOR $K = P$
(P XOR K) XOR $K = P$

There's no real security here. This kind of encryption is trivial to break, even without computers [360,832]. It will only take a few minutes with a computer.

Assume the plaintext is English. Furthermore, assume the key length is an arbitrary small number of bytes (although in the source code example, it is always eight bytes). Here's how to break it:

1. Discover the length of the key by a procedure known as counting coincidences [355]. Trying each byte displacement of the key against itself, count those bytes that are equal. If the two ciphertext portions have used the same key, something over 6% of the bytes will be equal. If they have used a different key, then less than 0.4% will be equal (assuming a random key encrypting normal ASCII text; other plaintext will have different numbers). The smallest displacement that indicates an equal key length is the length of the repeated key.
2. Shift the key by that length and XOR it with itself. This removes the key and leaves you with text XORed with itself. Since English has about one bit of real information per byte (see Section 9.1), there is plenty of redundancy for choosing a unique decryption.

Despite this, the list of software vendors that tout this sort of algorithm as being "almost as secure as DES" is staggering [774]. It might keep your kid sister from reading your files, but it won't stop a cryptographer for more than a few minutes.

1.2.4 One-time Pads

Believe it or not, there is a perfect encryption scheme. It's called a one-time pad and was invented in 1917 by Major Joseph Mauborgne and AT&T's Gilbert Vernam [462]. In its classical form, a one-time pad is nothing more than a large nonrepeating set of truly random key letters, written on sheets of paper and glued together in a pad. The sender uses each key letter on the pad to encrypt exactly one plaintext character. The receiver has an identical pad and uses each key on the pad, in turn, to decrypt each letter of the ciphertext.

Each key is used exactly once, for only one message. The sender encrypts the message and then destroys the pad's pages. The receiver does the same thing after decrypting the message. New message—new page and new key letters.

Assuming an adversary can't get access to the pages of the one-time pad used to encrypt the message, this scheme is perfectly secure. A given ciphertext message is equally likely to be any possible plaintext message of equal size. For example, if the message is:

UNETIMEPAD

and the key sequence from the pad is

TBFRGFARFM

then the ciphertext is

IPKLPSFHGQ

Since every key sequence is equally likely (remember, the keys are generated in a random manner), an adversary has no information with which to cryptanalyze the ciphertext. The key sequence could just as likely be:

POYYAEAAZX

which would decrypt to:

SALMONEGGS

or

BXFGBMTMXM

which would decrypt to:

GREENFLUID

This point bears repeating: since every plaintext message is equally possible, there is no way for the cryptanalyst to determine which plaintext message is the correct one. A random key sequence XORed with a nonrandom plaintext message produces a completely random ciphertext message, and no amount of computing power can change that.

The caveat, and this is a big one, is that the key letters have to be generated randomly. Any attacks against this scheme will be against the method used to generate the key sequence. If you use a cryptographically weak algorithm to

generate your key sequence, there might be trouble. If you use a real random source—this is much harder than it might first appear—it is safe.

Using a pseudo-random number generator doesn't count; often they have nonrandom properties. Many of the stream ciphers described later in this book try to approximate this system with a pseudo-random sequence, but most fail. The sequences they generate only seem random, but careful analysis yields nonrandomness, which a cryptanalyst can exploit. However, it is possible to generate real random numbers, even with a microcomputer. Some techniques for how to do this will be covered in Section 15.3.

The idea of a one-time pad can easily be extended to the encryption of binary data. Instead of a one-time pad consisting of letters, use a one-time pad of bits. Everything else remains the same, and the security is just as perfect.

This all sounds good, but there are a few problems. The length of the key sequence is equal to the length of the message. This might be suitable for a few short messages, but this will never work for a 1.44 Mbps communications channel. However, you can store 650 megabytes worth of random bits on a CD-ROM. This would make a perfect one-time pad for certain low-bandwidth applications, although then you have to deal with the problem of storing the CD-ROM when it is not in use and then destroying it once it has been completely used.

Even if you solve the key distribution and storage problem, you have to make sure the sender and receiver are perfectly synchronized. If the receiver is off by a bit, the message won't make any sense. On the other hand, if some bits are garbled during transmission, only those bits will be decrypted incorrectly.

One-time pads still have their applications in today's world, primarily for ultrasecure low-bandwidth channels. The hotline between the United States and the former Soviet Union was (is it still active?) rumored to be encrypted with a one-time pad. Many Soviet spy messages to agents were encrypted using one-time pads. These messages are still secure today and will remain that way forever. It doesn't matter how long the supercomputers work on the problem; it doesn't matter how many people may still be working on the problem half a century later with unimaginable machines and techniques. Even after the aliens from Andromeda land with their massive spaceships and undreamed-of computing power, they will not be able to read the Soviet spy messages encrypted with one-time pads (as long as the one-time pads used to generate the messages have been destroyed).

1.3 LARGE NUMBERS

Throughout this book use some large numbers to describe various cryptographic algorithms. It's easy to lose sight of these numbers and what they actually mean. Table 1.1 gives physical analogues for the kinds of numbers used in cryptography.

These numbers are order-of-magnitude estimates, and have been culled from a variety of sources. Many of the astrophysics numbers are explained in Freeman

TABLE 1.1
Large Numbers

Odds of being killed in an automobile accident (in the U.S. per year)	1 in 5600 (2^{13})
Odds of being killed in an automobile accident (in the U.S. per lifetime)	1 in 75 (2^{6})
Odds of drowning (in the U.S. per year)	1 in 59,000 (2^{16})
Time Until the Next Ice Age	14,000 (2^{14}) years
Time Until the Sun Goes Nova	10^{9} (2^{30}) years
Age of the Planet	10^{9} (2^{30}) years
Age of the Universe	10^{10} (2^{34}) years
If the Universe Is Closed:	
Total Lifetime of Universe	10^{11} (2^{37}) years
	10^{18} (2^{61}) seconds
If the Universe Is Open:	
Time Until Low-Mass Stars Cool Off	10^{14} (2^{47}) years
Time Until Planets Detach from Stars	10^{15} (2^{50}) years
Time Until Stars Detach from Galaxies	10^{19} (2^{64}) years
Time Until Orbits Decay by Gravitational Radiation	10^{20} (2^{67}) years
Time Until Black Holes Decay by the Hawking Process	10^{64} (2^{213}) years
Time Until All Matter Is Liquid at Zero Temperature	10^{65} (2^{216}) years
Time Until All Matter Decays to Iron	$10^{10^{26}}$ years
Time Until All Matter Collapses to Black Holes	$10^{10^{76}}$ years
Number of Atoms in the Planet	10^{51} (2^{170})
Number of Atoms in the Sun	10^{57} (2^{190})
Number of Atoms in the Galaxy	10^{67} (2^{223})
Number of Atoms in the Universe (Dark Matter Excluded)	10^{77} (2^{265})
Volume of the Universe	10^{84} (2^{280}) cm^3

Dyson's paper, "Time Without End: Physics and Biology in an Open Universe," in *Reviews of Modern Physics*, v. 52, n. 3, July 1979, pp. 447–460. Automobile accident deaths are calculated from the Department of Transportation's statistic of 178 road accidents per million people and an average lifespan of 75.4 years.

PART
ONE

Cryptographic Protocols

CHAPTER 2

Protocol Building Blocks

2.1 INTRODUCTION TO PROTOCOLS

The whole point of cryptography is to solve problems. (Actually, that's the whole point of computers—something many people tend to forget.) The types of problems that cryptography solves revolve around secrecy, distrustful and dishonest people, and privacy. You can learn all about algorithms and techniques, but these are merely interesting unless they solve problems. This is why we are going to look at protocols first.

A **protocol** is a series of steps, involving two or more parties, designed to accomplish a task. This is an important definition. A "series of steps" means that the protocol has a sequence, from start to finish. Every step must be executed in turn, and no step can be taken before the previous step is finished. "Involving two or more parties" means that at least two people are required to complete the protocol; one person alone does not make a protocol. Certainly one person can perform a series of steps to accomplish a task (e.g., baking a cake), but this is not a protocol. Finally, "designed to accomplish a task" means that the protocol must do something. Something that looks like a protocol but does not accomplish a task is not a protocol—it's a waste of time.

Protocols have other characteristics:

1. Everyone involved in the protocol must know the protocol and all of the steps to follow in advance.
2. Everyone involved in the protocol must agree to follow it.

3. The protocol must be unambiguous; each step must be well defined and there must be no chance of a misunderstanding.
4. The protocol must be complete; there must be a specified action for every possible situation.

The protocols in this book are organized as a series of steps. Execution of the protocol proceeds linearly through the steps, unless there are instructions to branch to another step. Each step involves at least one of two things: computations by one or more parties, or messages from one party to another.

Cryptographic Protocols

A **cryptographic protocol** is a protocol that uses cryptography. The parties involved can be friends and trust one another implicitly, or they can be adversaries and not trust one another at all. Of course, a cryptographic protocol involves some cryptographic algorithm, but generally the goal of the protocol is something beyond simple secrecy. The parties participating in the protocol might want to share parts of their secrets to compute a value, jointly generate a random sequence, convince one another of their identity, or simultaneously sign a contract. Cryptographic protocols that accomplish such goals have radically changed our ideas of what mutually distrustful parties can accomplish over a network.

The Purpose of Protocols

In daily life, there are informal protocols for almost everything: ordering goods over the telephone, playing poker, voting in an election. No one thinks much about these protocols; they have evolved over time, everyone knows how to use them, and they work.

More and more, people are communicating over computer networks instead of communicating face-to-face. Computers need formal protocols to do the same things that people do without thinking. If you moved from one state to another (or even from one country to another) and found a voting booth that looked completely different from the ones you were used to, you could easily adapt. Computers are not nearly so flexible.

Many face-to-face protocols rely on people's presence to ensure fairness and security. Would you send a stranger a pile of cash to buy groceries for you? Would you play poker with someone if you couldn't see him or her shuffle and deal? Would you mail the government your secret ballot without some assurance of anonymity?

It is naive to assume that the people on a computer network are going to be honest. It is naive to assume that the managers of a computer network are going to be honest. It is even naive to assume that the designers of a computer network were honest. Most will be honest, but it is the dishonest few we need to guard against. By formalizing protocols, we can examine ways in which dishonest parties can try to cheat and develop protocols that foil these cheaters.

In addition to formalizing behavior, protocols are useful because they abstract the process of accomplishing a task from the mechanism by which the task is accomplished. A communications protocol between two computers is the same whether the computers are IBM PCs, VAX computers, or facsimile machines. This abstraction allows us to examine the protocol for good features without getting bogged down in the implementation details. When we are convinced we have a good protocol, we can implement it in everything from computers to telephones to intelligent muffin toasters.

The Company

To help demonstrate the protocols, I have enlisted the aid of several people (see Table 2.1). Alice and Bob are the first two. They will perform all general two-person protocols. As a rule, Alice will initiate all protocols. Bob will play the second part. If the protocol requires a third or fourth person, Carol and Dave will perform those roles. Other actors will play specialized supporting roles; they will be introduced later.

Arbitrated Protocols

An **arbitrator** is a disinterested third party trusted to complete a protocol (see Figure 2.1a). Disinterested means that the arbitrator has no particular reason to complete the protocol and no particular allegiance to any of the people involved in the protocol. Trusted means that all people involved in the protocol accept that

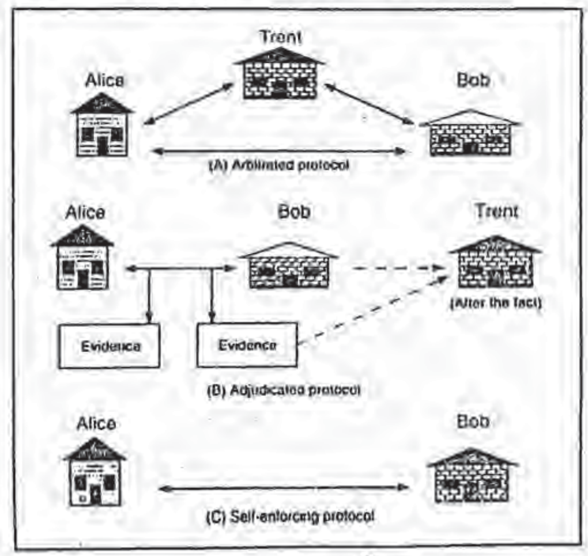


Figure 2.1
Types of protocols.

TABLE 2.1
Dramatis Personae

Alice	Participant in all the protocols.
Bob	Participant in the two-, three-, and four-party protocols.
Carol	Participant in the three- and four-party protocols.
Dave	Participant in the four-party protocols.
Eve	Eavesdropper.
Mallet	Malicious active attacker.
Trent	Trusted arbitrator.
Walter	Warden; he'll be guarding Alice and Bob in some protocols.
Peggy	Prover.
Victor	Verifier.

what is said is true, what is done is correct, and that his or her part of the protocol will be complete. Arbitrators can help complete protocols between two mutually distrustful parties.

In the real world, lawyers are often used as arbitrators. For example, Alice is selling a car to Bob, a stranger. Bob wants to pay by check, but Alice has no way of knowing if the check is good or not. Alice wants the check to clear before she turns the title over to Bob. Bob, who doesn't trust Alice any more than she trusts him, doesn't want to hand over a check without receiving a title.

Enter a lawyer trusted by both. With the help of the lawyer, Alice and Bob can agree on the following protocol to ensure that neither cheats the other:

(1) Alice gives the title and the keys to the lawyer.

(2) Bob gives the check to Alice.

(3) Alice deposits the check.

(4) After waiting a specified time period for the check to clear, the lawyer gives the title to Bob.

(5) If the check does not clear within the specified time period, Alice shows proof of this to the lawyer and the lawyer returns the title and the keys to Alice.

In this protocol, Alice trusts the lawyer not to give Bob the title and the keys unless the check has cleared and to give them back to her if the check does not clear. Bob trusts the lawyer to hold the title and the keys until the check clears before giving them to him. The lawyer doesn't care if the check clears or not. He will do what he is supposed to do in either case.

In the example, the lawyer is playing the part of an escrow agent. Escrow agents often arbitrate financial transactions. Lawyers act as arbitrators for wills and

sometimes for contract negotiations. The various stock exchanges act as arbitrators between buyers and sellers.

Bankers also arbitrate protocols. Bob can use a certified check to buy a car from Alice:

(1) Bob writes a check and gives it to the bank.

(2) After putting enough of Bob's money on hold to cover the check, the bank certifies the check and gives it back to Bob.

(3) Alice gives the title and the keys to Bob.

(4) Bob gives the certified check to Alice.

(5) Alice deposits the check.

This protocol works because Alice trusts the banker's certification. Alice trusts the bank to hold her money and not to use it to finance shaky real estate operations in mosquito-infested countries.

A notary public is another arbitrator. When Bob receives a notarized document from Alice, he is convinced that Alice signed the document voluntarily and with her own hand. The notary can, if necessary, stand up in court and attest to that fact.

The concept of an arbitrator is as old as society. There have always been people—rulers, priests, and so on—who have the authority to act fairly. Arbitrators have a certain social role and position in our society; betraying the public trust would jeopardize that. Lawyers who play games with escrow accounts face almost-certain disbarment, for example. This rosy picture doesn't always exist in the real world, but it's the ideal.

This ideal can translate to the computer world, but there are several problems with translating arbitrators into computers:

- It is easier to find and trust a neutral third party if you know who the party is, can see his face, or have a feeling that he is a real person. Two parties suspicious of each other are also likely to be suspicious of some faceless arbitrator somewhere else on the network.
- The computer network must bear the cost of maintaining an arbitrator. We all know what lawyers charge; who wants to bear that kind of network overhead?
- There is a delay inherent in any arbitrated protocol. The arbitrator must deal with every transaction.
- Arbitrators are potential bottlenecks in large-scale implementations of any protocol. Increasing the number of arbitrators in the implementation can mitigate this problem, but then the cost increases.
- Since everyone on the network must trust the arbitrator, he represents a vulnerable point for anyone trying to subvert the network.

Even so, arbitrators still have a role to play. In protocols using a trusted arbitrator, the part will be played by Trent.

Adjudicated Protocols

Because of the high cost of hiring arbitrators, arbitrated protocols can be subdivided into two lower-level subprotocols. One is a nonarbitrated subprotocol, executed every time people want to complete the protocol. The other is an arbitrated subprotocol, executed only in exceptional circumstances—when there is a dispute. This special type of arbitrator is called an **adjudicator** (see Figure 2.1b).

An adjudicator is also a disinterested and trusted third party. Unlike an arbitrator, he is not directly involved in every protocol. The adjudicator is called in only to determine whether a transaction was performed fairly.

Judges are professional adjudicators. Unlike a notary public, a judge is brought in only if there is a dispute. Alice and Bob can enter into a contract without a judge. A judge never sees the contract unless one of them hauls the other into court.

This contract-signing protocol can be formalized in this way:

Nonarbitrated subprotocol (executed every time):

(1) Alice and Bob negotiate the terms of the contract.

(2) Alice signs the contract.

(3) Bob signs the contract.

Adjudicated subprotocol (executed only in case of a dispute):

(1) Alice and Bob appear before a judge.

(2) Alice presents her evidence.

(3) Bob presents his evidence.

(4) The judge rules on the evidence.

The key difference between an adjudicator and an arbitrator (as I use the terms in this book) is that the adjudicator is not always necessary. If there is a dispute, a judge is called in to adjudicate. If there is no dispute, using a judge is unnecessary.

There are adjudicated computer protocols. These protocols rely on the involved parties to be honest; but if someone cheats, there is a body of data collected so that a disinterested third party could determine if someone cheated. In a good adjudicated protocol, the adjudicator could also determine the cheater's identity. In real life, adjudicators are seldom called. The inevitability of detection discourages cheating, and people remain honest.

Self-Enforcing Protocols

A **self-enforcing protocol** is the best type of protocol. The protocol itself guarantees fairness (see Figure 2.1c). No arbitrator is required to complete the protocol. No

adjudicator is required to resolve disputes. The protocol is constructed so that there cannot be any disputes. If one of the parties tries to cheat, the other party immediately detects the cheating and the protocol stops. Whatever the cheating party hoped would happen by cheating doesn't happen.

In the best of all possible worlds, every protocol would be self-enforcing. Unfortunately, there is not a self-enforcing protocol for every situation. I'll do my best, though.

Attacks Against Protocols

Attacks against protocols can be directed against the cryptographic algorithms used in the protocol, the cryptographic techniques used to implement the algorithm (e.g., key generation), or the protocol itself. This section of the book discusses only the protocols. For the time being we will assume that the cryptographic algorithms and techniques are secure, and look only at attacks against the protocols themselves.

There are various ways people can try to attack a protocol. Someone not involved in the protocol can eavesdrop on some or all of the protocol. This is called a **passive attack**, because the attacker does not affect the protocol. All he can do is observe the protocol and attempt to gain information. This kind of attack corresponds to a ciphertext-only attack, as discussed in Section 1.1. In these protocols, the part of the eavesdropper will be played by Eve.

Alternatively, an attacker could try to alter the protocol to his own advantage. He could introduce new messages in the protocol, delete existing messages, substitute one message for another, destroy a communications channel, or alter stored information in a computer. These are called **active attacks**, because they require active intervention.

Passive attackers are concerned solely with obtaining information about the parties involved in the protocol. They do this by collecting the messages passing among various parties and attempting to cryptanalyze them. Active attacks, on the other hand, can have much more diverse objectives. The attacker could be interested in obtaining information, degrading system performance, corrupting existing information, or gaining unauthorized access to resources.

Active attacks are much more serious, especially in protocols in which the different parties don't necessarily trust one another. The attacker does not have to be a complete outsider. He could be a legitimate system user. There could even be many active attackers, all working together, each of them a legitimate system user. The part of the malicious active attacker will be played by Mallet.

It is also possible that the attacker could be one of the parties involved in the protocol. He may lie during the protocol or not follow the protocol at all. This type of attacker is called a **cheater**. **Passive cheaters** follow the protocol but try to obtain more information than the protocol intends them to. **Active cheaters** disrupt the protocol in progress in an attempt to cheat.

It is very difficult to maintain a system's security if most of the parties involved are active cheaters, but sometimes it is possible for legitimate parties to detect that active cheating is going on. Certainly, protocols should be secure against passive cheating.

2.2 COMMUNICATIONS USING SYMMETRIC CRYPTOGRAPHY

How do two parties communicate securely? They encrypt their communications, of course. The complete protocol is more complicated than that. Let's look at what must happen for Alice to send an encrypted message to Bob.

(1) Alice and Bob agree on a cryptosystem.

(2) Alice and Bob agree on a key.

(3) Alice takes her plaintext message and encrypts it using the encryption algorithm and the key. This creates a ciphertext message.

(4) Alice sends the ciphertext message to Bob.

(5) Bob decrypts the ciphertext message with the same algorithm and key and reads it.

What can Eve, an eavesdropper sitting between Alice and Bob, learn from listening in on this protocol? If all she hears is the transmission in step (4), she must try to cryptanalyze the ciphertext. This passive attack is a ciphertext-only attack; there are an array of algorithms that are resistant (as far as we know) to whatever computing power Eve could bring to bear on the problem.

Eve isn't stupid, though. She knows that if she can listen in on steps (1) and (2), she's succeeded. She would know the algorithm and the key; she would know just as much as Bob. When the message comes across the communications channel in step (4), all she has to do is decrypt it herself.

This is why key management is such an important matter in cryptography. A good cryptosystem is one in which all the security is inherent in the key and none is inherent in the algorithm. With a symmetric algorithm, Alice and Bob can perform step (1) in public, but they must perform step (2) in secret. The key must remain secret before, during, and after the protocol; otherwise the message will no longer be secure. (Public-key cryptography solves this problem another way and will be discussed in Section 2.5.)

Mallet, an active attacker, could do a few other things. He could attempt to break the communications path in step (4), ensuring that Alice could not talk to Bob at all. Mallet could also intercept Alice's messages and substitute ones of his own. If he also knew the key (by intercepting the communication in step (2), or by breaking the cryptosystem), he could encrypt his own message and send it to Bob in place of the intercepted message. Bob would have no way of knowing that the message had not come from Alice. If Mallet didn't know the key, he could only create a replacement message that would decrypt to gibberish. Bob, thinking the message came from Alice, might conclude that either the network or Alice had some serious problems.

What about Alice? What can she do to disrupt the protocol? She can give a copy of the key to Eve. Now Eve can read whatever Bob says. Bob, who has no idea that Eve has the key, thinks he is talking securely to Alice. He has no idea Eve is reprinting his words in the *New York Times*. Although serious, this is not a problem with the protocol. There is nothing to stop Alice from giving Eve a copy of the plaintext at any point during the protocol. Of course, Bob could also do anything that Alice could. This protocol assumes that Alice and Bob trust each other.

In summary, symmetric cryptosystems have the following problems:

- If the key is compromised (stolen, guessed, extorted, bribed, etc.), then the adversary who had the key can decrypt all message traffic encrypted with that key. He or she can also pretend to be one of the parties and produce false messages to fool the other party. It is very important to change keys frequently to minimize this problem.
- Keys must be distributed in secret. They are more valuable than any of the messages they encrypt, since knowledge of the key means knowledge of all the messages. For encryption systems that span the world, this can be a daunting task. Often couriers hand-carry keys to their destinations.
- Assuming a separate key is used for each pair of users in a network, the total number of keys increases rapidly as the number of users increases. For example, 10 users need 45 different keys to talk with one another; 11 users need 55 different keys. This problem can be minimized by keeping the number of users small, but that is not always possible.

2.3 ONE-WAY FUNCTIONS

The notion of one-way functions is central to public-key cryptography. While not a protocol in itself, one-way functions are a fundamental building block for most of the protocols described in this book.

A one-way function is a function that is relatively easy to compute but significantly harder to undo or reverse. That is, given x it is easy to compute $f(x)$, but given $f(x)$ it is hard to compute x. In this context, "hard" means, in effect, that it would take millions of years to compute the function even if all the computers in the world were assigned to the problem.

Taking a watch apart is a good example of a one-way function. It is easy to smash a watch into hundreds of tiny pieces. However, it's not easy to put all of those tiny pieces back together into a functional watch.

This sounds accurate, but in fact it isn't demonstratively true. If we are being strictly mathematical, there is no proof that one-way functions exist, nor is there any real evidence that they can be constructed [138,322,366,393]. Even so, there are many functions that look and smell one-way: we can compute them efficiently and, as of now, know of no easy way to reverse them. For example, x^2 is easy to compute, but $\sqrt{x}$ is much harder. For the rest of this section, I'm going to pretend that there are one-way functions. I'll talk more about this in Section 9.2.

So, what good are one-way functions? I can't use them for encryption as is. A message encrypted with the one-way function isn't useful; no one could decrypt it. (Exercise: Write a message on a plate, smash the plate into tiny bits, and then give the bits to a friend. Ask the friend to read the message. Observe how impressed your friend is with the one-way function.) For encryption, we need something called a **trap-door one-way function**. (There are cryptographic applications for one-way functions; see Section 3.2.)

A **trap-door one-way function** is a special type of one-way function with a secret trap door. It is easy to compute in one direction and hard to compute in the other direction. But, if you know the secret, you can easily compute the function in the other direction. That is, it is easy to compute $f(x)$ given x, and hard to compute x given $f(x)$. However, there is some secret information, y, such that given $f(x)$ and y it is easy to compute x. In our watch example, the secret information might be a set of assembly instructions for the watch.

A mailbox is a good example of a trap-door one-way function. Anyone can easily put mail into the box; just open the slot and drop it in. Putting mail in a mailbox is a public activity. Opening the mailbox is not a public activity. It's hard; you would need welding torches or other tools. However, if you have the secret (the key or the combination), it's easy to open the mailbox. Public-key cryptography is a lot like that.

2.4 ONE-WAY HASH FUNCTIONS

A **one-way hash function** has many names: **compression function**, **contraction function**, **message digest**, **fingerprint**, **cryptographic checksum**, **data integrity check (DIC)**, **manipulation detection code (MDC)**, **message authentication code (MAC)**, and **data authentication code (DAC)**. Whatever you call it, it is central to modern cryptography. One-way hash functions are another building block for many protocols.

Hash functions have been used in computer science for a long time. A hash function is a function, mathematical or otherwise, that takes an input string and converts it to a fixed-size (often smaller) output string. A simple hash function would be a function that takes an input string and returns a byte consisting of the XOR of all the input bytes. The whole point here is to fingerprint the input string: to produce a value that can indicate whether a candidate string is likely to be the same as the input string. Because hash functions are typically many to one, we cannot use them to determine with certainty that the two strings are equal, but we can use them to get a reasonable assurance of equality.

A one-way hash function is a hash function that is also a one-way function: it is easy to compute a hash value from an input string, but it is hard to generate a string that hashes to a particular value. The hash function in the previous paragraph is not one-way: given a particular byte value, it is trivial to generate a string of bytes whose XOR is that value. You can't do that with a one-way hash function.

A particular one-way hash function may return values on the order of 128 bits long, so that there are 2^{128} possible hashes. The number of trials required to find a random string with the same hash value as a given string is 2^{128}, and the number of trials required to find two random strings having the same (random) hash value is 2^{64}.

There are two primary types of one-way hash functions: those with a key and those without a key. One-way hash functions without a key can be calculated by anyone; the hash is solely a function of the input string. One-way hash functions with a key are a function of both the input string and the key; only someone with the key can calculate the hash value. (It's the same as calculating the one-way hash and then encrypting it.)

Algorithms specifically designed to be one-way hash functions have been developed (see Chapter 14). These are pseudo-random functions; any hash value is equally likely. The output is not dependent on the input in any discernible way. A single change in any input bit changes, on the average, half the output bits. Given a hash value, it is computationally unfeasible to find an input string that hashes to that value.

Think of it as a way of fingerprinting files. If you want to verify that someone has a particular file (that you also have), but you don't want her to send it to you, then ask her for the hash value. If she sends you the correct hash value, then it is almost certain that she has that file. Normally, you would use a one-way hash function without a key, so that anyone can verify the hash. If you only want the recipient to be able to verify the hash, then use a one-way hash function with a key.

2.5 COMMUNICATIONS USING PUBLIC-KEY CRYPTOGRAPHY

Think of a symmetric algorithm as a safe. The key is the combination. Someone with the combination can open the safe, put a document inside, and close it again. Someone else with the combination can open the safe and take the document out. Anyone without the combination is forced to learn safecracking.

In 1976, Whitfield Diffie and Martin Hellman changed that paradigm of cryptography forever [299]. They described **public-key cryptography**. Instead of one key, there are two different keys: one public and the other private. Moreover, it is computationally unfeasible to deduce the private key from the public key. Anyone with the public key (which, presumably, is public) can encrypt a message but not decrypt it. Only the person with the private key can decrypt the message. It is as if someone cut a mail slot into the cryptographic safe. Anyone can slip messages into the slot, but only someone with the private key can open the safe and read the messages.

Mathematically, the process is based on the trap-door one-way functions discussed above. Encryption is the easy direction. Instructions for encryption are the public key: anyone can encrypt a message. Decryption is the hard direction. It's made hard enough that people with Cray computers and thousands (even millions) of years couldn't decrypt the message without the secret. The secret, or trap door, is the private key. With that secret, decryption is as easy as encryption.

This is how Alice can send a message to Bob using public-key cryptography:

(1) Alice and Bob agree on a public-key cryptosystem.

(2) Bob sends Alice his public key.

public key, he decrypts it with his private key, re-encrypts it with Bob's public key, and sends it on to Bob.

(4) When Bob sends a message to Alice, encrypted in "Alice's" public key, Mallet intercepts it. Since the message is really encrypted with his own public key, he decrypts it with his private key, re-encrypts it with Alice's public key, and sends it on to Alice.

Even if Alice and Bob's public keys are stored on a database, this attack will work. Mallet can intercept Alice's database inquiry, and substitute his own public key for Alice's. He can do the same to Bob. He can break into the database surreptitiously and substitute his key for Alice's and Bob's. The next day he simply waits for Alice and Bob to talk with each other. Then he intercepts and modifies the messages, and he has succeeded.

This **man-in-the-middle attack** works because Alice and Bob have no way to verify that they are talking to each other. Assuming Mallet is quick and doesn't cause any noticeable network delays, the two of them have no idea that someone sitting between them is reading all of their supposedly secret communications.

The easiest way to protect against this is for Alice and Bob to confirm that they are using the same key. If they can communicate via voice (or another way that allows them to unambiguously identify each other), they can each read a one-way hash of their key to the other. Of course, this isn't always possible.

Interlock Protocol

The **interlock protocol**, invented by Ron Rivest and Adi Shamir [748], has a good chance of foiling the man-in-the-middle attack. Here's how it works.

(1) Alice sends Bob her public key.

(2) Bob sends Alice his public key.

(3) Alice encrypts her message using Bob's public key. She sends half of the encrypted message to Bob.

(4) Bob encrypts his message using Alice's public key. He sends half of the encrypted message to Alice.

(5) Alice sends the other half of her encrypted message to Bob.

(6) Bob puts the two halves of Alice's message together and decrypts it with his private key. Bob sends the other half of his encrypted message to Alice.

(7) Alice puts the two halves of Bob's message together and decrypts it with her private key.

The important point is that half of the message is useless without the other half; it can't be decrypted. Bob cannot read any part of Alice's message until step (6); Alice cannot read any part of Bob's message until step (7). There are a number of ways to do this:

- If the encryption algorithm is a block algorithm, half of each block (for example, every other bit) could be sent in each half message.
- Decryption of the message could be dependent on an initialization vector (see Section 8.1.3), which would be sent with the second half of the message.
- The first half of the message could be a one-way hash function of the encrypted message (see Section 2.4), and the encrypted message itself could be the second half.

To see how this causes a problem for Mallet, let's review his attempt to subvert the protocol. He can still substitute his own public keys for Alice's and Bob's in steps (1) and (2). But now, when he intercepts half of Alice's message in step (3), he cannot decrypt it with his private key and re-encrypt it with Bob's public key. He has to invent a totally new message and send half of it to Bob. When he intercepts half of Bob's message to Alice in step (4), he has the same problem. He cannot decrypt it with his private key and re-encrypt it with Alice's public key. He has to invent a totally new message and send half of it to Alice. By the time he intercepts the second halves of the real messages in steps (5) and (6), it is too late for him to change the new messages he invented. The conversation between Alice and Bob will necessarily be completely different.

Mallet could possibly get away with this scheme. If he knows Alice and Bob well enough to mimic both sides of a conversation between them, they might never realize that they are being duped. But surely this is much harder than sitting between the two of them, intercepting and reading their messages.

Key Exchange with Digital Signatures

Implementing digital signatures during a session-key exchange protocol circumvents this man-in-the-middle attack as well. A KDC signs both Alice's and Bob's public keys. The signed keys include a signed certification of ownership. When Alice and Bob receive the keys, they each verify the KDC's signature. Now they know that the public key belongs to that other person. The key exchange protocol can then proceed.

Mallet has serious problems. He cannot impersonate either Alice or Bob because he doesn't know either of their private keys. He cannot substitute his public key for either of theirs because it isn't signed by the KDC. All he can do is listen to the encrypted traffic go back and forth or disrupt the lines of communication and prevent Alice and Bob from talking.

This protocol also uses a KDC, but the risk of compromising the KDC is much less. If Mallet breaks into the KDC, all he gets is the KDC's private key. This key enables him only to sign new keys; it does not let him decrypt any session keys or read any message traffic. To be able to read the traffic, Mallet has to be able to impersonate a user on the network and trick legitimate users into encrypting messages with his phony public key.

Mallet can launch that kind of attack. With the KDC's private key, he can create phony signed keys to fool both Alice and Bob. Then, he can either exchange them

(3) Alice encrypts her message using Bob's public key and sends it to Bob.

(4) Bob then decrypts Alice's message using his private key.

Notice how public-key cryptography solves the primary problem with symmetric cryptosystems: by distributing the key. With a conventional cryptosystem, Alice and Bob have to agree on the same key. Alice could choose one at random, but she still has to get it to Bob. She could hand it to him sometime beforehand, but that requires foresight. She could send it to him by registered mail, but that takes time. With public-key cryptography, there is no problem. With no prior arrangements, Alice can send a secure message to Bob. Eve, listening in on the entire exchange, has Bob's public key and a message encrypted in that key, but cannot recover either Bob's private key or the message.

More commonly, a network of users agrees on a public-key cryptosystem. Every user has his or her own public key and private key, and the public keys are all published in a database somewhere. Now the protocol is even easier:

(1) Alice gets Bob's public key from the database.

(2) Alice encrypts her message using Bob's public key and sends it to Bob.

(3) Bob then decrypts Alice's message using his private key.

In the first protocol, Bob had to send Alice his public key before she could send him a message. The second protocol is more like traditional mail. Bob is not involved in the protocol until he wants to read his message.

Attacks Against Public-Key Cryptography

In all these public-key digital signature protocols, I glossed over how Alice gets Bob's public key. Section 3.1 discusses this in detail, but it is worth mentioning here.

The obvious way to exchange a public key is from a secure database somewhere. The database has to be public, so that anyone can get anyone else's public key. The database also has to be protected from access by anyone except Trent; otherwise Mallet could substitute a key of his choosing for Alice's. After he did that, Bob couldn't read his messages, but Mallet could.

Even if the public keys are stored in a secure database, Mallet could still substitute one for another during transmission. To prevent this, Trent can sign each of the public keys with his own private key. Trent, when used in this manner, is often known as a **Key Certification Authority** or **Key Distribution Center (KDC)**. In practical implementations, the KDC signs a compound message consisting of the user's name, the public key, and any other important information about the user. This signed compound message is stored in the KDC's database. When Alice gets Bob's key, she verifies the KDC's signature to assure herself of the key's validity.

In the final analysis, this is not making things impossible for Mallet, only more difficult. Alice still has the KDC's public key stored somewhere. Mallet has to substitute that key for his own public key, corrupt the database, and substitute the valid keys with his own keys, all signed with his private key as if he were the KDC, and then he's in business. But even paper-based signatures can be forged if Mallet goes to enough trouble. As mentioned earlier, this will be discussed in minute detail in Section 3.1.

Hybrid Cryptosystems

Public-key algorithms are significantly slower than symmetric algorithms; symmetric algorithms are generally at least 1000 times faster than public-key algorithms. In most practical implementations public-key cryptography is used to secure and distribute session keys, and those session keys are used with symmetric algorithms to secure message traffic [499]. This is sometimes called a hybrid cryptosystem (see Section 3.1).

2.6 DIGITAL SIGNATURES

Handwritten signatures on documents have long been used as proof of authorship of, or at least agreement with, the contents of the document. What is it about a signature that is so compelling?

1. The signature is unforgeable. The signature is proof that the signer deliberately signed the document.
2. The signature is authentic. The signature convinces the document's recipient that the signer deliberately signed the document.
3. The signature is not reusable. The signature is part of the document, and an unscrupulous person cannot move the signature to a different document.
4. The signed document is unalterable. After the document is signed, it cannot be altered.
5. The signature cannot be repudiated. The signature and the document are physical things. The signer cannot later claim that he or she didn't sign it.

In reality, none of these statements about signatures is completely true. Signatures can be forged; signatures can be lifted from one piece of paper and moved to another. Documents can be altered after signing. However, we are willing to live with these problems because of the difficulty to cheat and the risk of detection.

We would like to do this sort of thing on computers, but there are problems. First, bit streams are easy to copy. Even if a person's signature were difficult to forge (a graphical image of a written signature, for example), it is easy to move a valid signature from one document to another document. The mere presence of such a signature means nothing. Second, documents are easy to modify after they are signed, without leaving any evidence of modification.

Signing Documents with Symmetric Cryptosystems and an Arbitrator

Alice wants to sign a digital message and send it to Bob. With the help of Trent and a symmetric cryptosystem, she can:

Trent is a powerful, trusted arbitrator. He can communicate with both Alice and Bob (and everyone else who may want to sign a digital document). He shares a secret key, K_A, with Alice, and a different secret key, K_B, with Bob. These keys have been established long before the protocol begins and can be reused multiple times for multiple signings.

(1) Alice encrypts her message to Bob with K_A and sends it to Trent.

(2) Trent decrypts the message with K_A.

(3) Trent takes the decrypted message, a statement that he has received this message from Alice, and a copy of Alice's encrypted message. He encrypts the whole bundle with K_B.

(4) Trent sends the encrypted bundle to Bob.

(5) Bob decrypts the bundle with K_B. He can now read both the message and Trent's certification that Alice sent it.

How does Trent know that the message is from Alice, and not from some imposter? He infers it from the message's encryption. Since only he and Alice share their secret key, only Alice could encrypt a message using it.

Is this as good as a paper signature? Let's look at the characteristics we want:

1. This signature is unforgeable. Only Alice (and Trent—but everyone trusts him) knows K_A, so only Alice could have sent Trent a message encrypted with K_A. If someone tried to impersonate Alice, Trent would have immediately realized this in step (2) and would not send the message to Bob.
2. This signature is authentic. Trent is a trusted arbitrator, and Trent knows that the message came from Alice. Trent's certification is proof enough.
3. This signature is not reusable. If Bob tried to take Trent's certification and attach it to another message, Alice would cry foul. An arbitrator (it could be Trent, or it could be a completely different arbitrator with access to the same information) would ask Bob to produce both the message and Alice's encrypted message. The arbitrator would then encrypt the message with K_A and notice that it did not match the encrypted message that Bob gave him. Bob, of course, could not produce an encrypted message that does match because he does not know K_A.
4. The signed document is unalterable. Were Bob to try to alter the document after receipt, Trent could prove foul play in exactly the same manner described above.
5. The signature cannot be repudiated. Even if Alice later claims that she never sent the message, Trent's certification says otherwise. Remember, Trent is trusted by everyone; what he says is true.

If Bob wants to show Carol a document signed by Alice, he can't reveal his secret key to her. He has to go through Trent again:

(1) Bob takes the message and the statement that the message came from Alice, encrypts them with K_B, and sends them back to Trent.

(2) Trent decrypts the bundle with K_B.

(3) Trent re-encrypts the bundle with the secret key he shares with Carol, K_C, and sends it to Carol.

(4) Carol decrypts the bundle with K_C. She can now read both the message and Trent's certification that Alice sent it.

These protocols work, but they're time-consuming for Trent. He has to spend his days decrypting and encrypting messages, acting as the intermediary between every pair of people who want to send signed documents to one another. He is going to be a bottleneck in any communications system, even if he's a mindless software program.

Harder still is creating and maintaining someone like Trent, someone that everyone on the network trusts. Trent has to be infallible; even if he makes one mistake in a million signatures, no one is going to trust him. Trent has to be completely secure. If his database of secret keys ever gets out, or if someone manages to modify his code, everyone's signatures would be completely useless. False documents purported to be signed years ago could appear. Chaos would result. This might work in theory, but it doesn't work very well in practice.

Signing Documents with Public-Key Cryptography

There are public-key algorithms that can be used for digital signatures. In some algorithms—RSA (see Section 12.4) is an example—either the public key or the private key can be used for encryption. Encrypt a document using your private key and you have a secure digital signature. In other cases—DSA (see Section 13.5) is an example—there is a separate algorithm for digital signatures that cannot be used for encryption. This idea was first invented by Diffie and Hellman [299] and further expanded and elaborated on in other texts [723,749,570,724].

The basic protocol is simple:

(1) Alice uses a digital signature algorithm with her private key to sign the message.

(2) Alice sends the document to Bob.

(3) Bob uses the digital signature algorithm with Alice's public key to verify the signature.

This protocol is far better than the previous one. Trent is not necessary; Alice and Bob can do it by themselves. Bob does not even need Trent to resolve disputes: if he cannot perform step (3), then he knows the signature is not valid.

This protocol also satisfies the characteristics we're looking for:

1. The signature is unforgeable; only Alice knows her private key.
2. The signature is authentic; when Bob verifies the message with Alice's public key, he knows that she signed it.
3. The signature is not reusable; the signature is a function of the document and cannot be transferred to any other document.
4. The signed document is unalterable; if there is any alteration to the document, it can no longer be verified with Alice's public key.
5. The signature cannot be repudiated. Bob doesn't need Alice's help to verify her signature.

Signing Documents and Timestamps

Actually, Bob can cheat Alice in certain circumstances. He can reuse the signature and the document together. This isn't exciting if Alice signed a contract (what's another copy of the same contract, more or less?), but it can be very exciting if Alice signed a digital check.

Let's say Alice sends Bob a signed digital check for $100. Bob takes the check to the bank, which verifies the signature and moves the money from one account to the other. Bob, who is an unscrupulous character, saves a copy of the digital check. The following week, he again takes it to the bank (or maybe to a different bank). The bank verifies the signature and moves the money from one account to the other. If Alice never balances her checkbook, Bob can keep this up forever.

Consequently, digital signatures often include timestamps. The date and time of the signature are attached to the message and signed along with the rest of the message. The bank stores this timestamp in a database. Now, when Bob tries to cash Alice's check a second time, the bank checks the timestamp against its database. Since the bank already cashed a check from Alice with the same timestamp, the bank calls the police. Bob then spends fifteen years in Leavenworth Prison reading up on cryptographic protocols.

Signing Documents with Public-Key Cryptography and One-Way Hash Functions

In practical implementations, public-key algorithms are often too inefficient to encrypt long documents. To save time, digital signature protocols are often implemented with one-way hash functions [246,247]. Instead of signing a document, Alice signs the hash of the document. In this protocol, both the one-way hash function and the digital signature algorithm are agreed upon beforehand.

(1) Alice produces a one-way hash of a document.

(2) Alice signs the hash with her private key, thereby signing the document.

(3) Alice sends the document and the signed hash to Bob.

(4) Bob produces a one-way hash of the document that Alice sent. He then verifies the signed hash with Alice's public key and compares it with the hash he generated. If they match, the signature is valid.

Speed increases drastically, and, since the chances of two different documents having the same 160-bit hash are only one in 2^{160}, anyone can safely equate a signature of the hash with a signature of the document. If a non-one-way hash function were used, it would be an easy matter to create multiple documents that hashed to the same value, so that anyone signing a particular document would be duped into signing a multitude of documents. This protocol cannot work without one-way hash functions.

This protocol has other benefits. First, the signature has to be kept separate from the document. Second, the recipient's storage requirements for the document and signature are much smaller.

An archival system can use this type of protocol to verify the existence of documents without storing their contents. The central database could just store the hashes of files. It doesn't have to see the files at all; users submit their hashes to the database, and the database timestamps the submissions and stores them. If there is any disagreement in the future about who created a document and when, the database could resolve it by finding the hash in its files. This has vast implications concerning privacy: Alice could copyright a document but still keep the document secret. Only if she wished to prove her copyright would she have to make the document public. (See Section 3.8.)

Algorithms and Terminology

There are many digital signature algorithms. All of them are public-key algorithms: there is some secret information that only allows someone who knows the information to sign documents, and there is some public information that allows everyone to verify documents. Sometimes the signing process is called encrypting with a private key, and the verification process is called decrypting with a public key. This is misleading and is only true for one algorithm. Additionally, there are often implementation differences. For example, one-way hash functions and timestamps sometimes add extra steps to the process of signing and verifying. Many algorithms can be used for digital signatures, but not for encryption.

In general, I will refer to the signing and verifying processes without any details of the algorithms involved. Signing a message with private key K is:

$$S_K(M)$$

and verifying a message with the corresponding public key is:

$$V_K(M)$$

The bit string attached to the document when signed (in the above example, the one-way hash of the document encrypted with the private key) will be called the **digital signature**, or just the **signature**. The entire protocol, by which the receiver of a message is convinced of the identity of the sender and the integrity of the message, is called **authentication**. Further details on these protocols are in Section 3.2.

Multiple Signatures

How could Alice and Bob sign the same digital document? Without one-way hash functions, there are two options. In the first option, Alice and Bob sign separate copies of the document itself. The resultant message would be twice the size of the original document—this is not optimal. In the second option, Alice would sign the document first and then Bob would sign Alice's signature. This works, except it is impossible to verify Alice's signature without also verifying Bob's.

If a one-way hash of the document is signed instead of the document itself, multiple signatures are easy:

(1) Alice signs the document.

(2) Bob signs the document.

(3) Bob sends his signature to Alice.

(4) Alice or Bob sends the document, her signature, and his signature to Carol.

(5) Carol verifies both Alice's signature and Bob's signature.

Alice and Bob can do steps (1) and (2) either in parallel or in series. In step (5), Carol can verify one signature without having to verify the other.

Cheating with Digital Signatures

Alice can cheat with digital signatures, and there's nothing that can be done about it. She wishes to sign a document and then later claim that she did not. First, she signs the document normally. Then, she anonymously publishes her private key, conveniently loses it in some public place, or just pretends to do either of the above. Now, anyone who finds the key can pretend to be Alice and sign documents. Alice then claims that her signature has been compromised and that others are using it, pretending to be her. She disavows signing the document and any others that she signed using that private key. Timestamps can limit the effects of this kind of cheating, but Alice can always claim that her key was compromised earlier. If Alice times things well, she can sign a document and then successfully claim that she didn't. This is why one hears so much about private keys buried in tamper-resistant modules—so that Alice can't get at hers and abuse it.

Although there is nothing that can be done about this possible abuse, one can take steps to guarantee that old signatures are not invalidated by actions taken in disputing new ones. (For example, Alice could "lose" her key to prevent her from paying Bob for the junker he sold her yesterday and in the process invalidate her year-old mortgage.) The solution is for the receiver of a signed document to have it timestamped by an arbitrator. This timestamp proves that the document was signed at a given time. Now, when Alice either pretends to lose or actually loses her key, only documents signed after she reports the loss are considered invalid. This is similar to the rules about reporting a stolen credit card.

Applications of Digital Signatures

One of the earliest proposed applications of digital signatures was to facilitate the verification of nuclear test ban treaties [817]. The United States and the former Soviet Union can put seismometers on each other's soil to monitor each other's nuclear tests. The problem is that the monitoring nation must assure itself that the host nation is not tampering with the data from the monitoring nation's seismometers. Simultaneously, the host nation wants to assure itself that the monitor is sending only the specific information needed for monitoring. Conventional authentication techniques can solve the first problem, but only digital signatures can solve both problems. The host nation can read, but not alter, data from the seismometer; and the monitoring nation knows that the data has not been tampered with.

2.7 DIGITAL SIGNATURES WITH ENCRYPTION

By combining digital signatures with public-key cryptography, we can develop a protocol that combines the security of encryption with the authenticity of digital signatures. For example, think of a signed letter in an envelope: the digital signature provides proof of authorship, and encryption provides privacy.

(1) Alice signs the message with her private key.

$$S_A(M)$$

(2) Alice encrypts the signed message with Bob's public key and sends it to Bob.

$$E_B(S_A(M))$$

(3) Bob decrypts the message with his private key.

$$D_B(E_B(S_A(M))) = S_A(M)$$

(4) Bob verifies with Alice's public key and recovers the message.

$$V_A(S_A(M)) = M$$

Of course, timestamps should be used with this protocol to prevent reuse of messages. Timestamps can also protect against other potential pitfalls, such as the one described in the next section.

Resending the Message as a Receipt

Consider an implementation of this protocol, with the additional feature of confirmation messages. Whenever someone receives a message, he or she sends it back to the sender as a confirmation of receipt.

(1) Alice signs a message with her private key, encrypts it with Bob's public key, and sends it to Bob.

$E_B(S_A(M))$

(2) Bob decrypts the message with his private key and verifies the signature with Alice's public key, thereby verifying that Alice signed the message and recovering the message.

$V_A(D_B(E_B(S_A(M)))) = M$

(3) Bob signs the message with his private key, encrypts it with Alice's public key, and sends it back to Alice.

$E_A(S_B(M))$

(4) Alice decrypts the message with her private key, and verifies the signature with Bob's public key. If the resultant message is the same one she sent to Bob, she knows that Bob received the message accurately.

If the same algorithm is used for both encryption and digital signatures there is a possible attack [305]. In these cases, the digital signature operation is the inverse of the encryption operation: $V_X = E_X$ and that $S_X = D_X$.

Assume that Mallet is a legitimate system user with his own public and private key. Now, let's watch as he reads Bob's mail. First, he records Alice's message to Bob in step (1). Then, at some later time, he sends that message to Bob, claiming that it came from him (Mallet). Bob thinks that it is a legitimate message from Mallet, so he decrypts the message with his private key and then tries to verify Mallet's signature by decrypting it with Mallet's public key. The resultant message, which is pure gibberish, is:

$E_M(D_B(E_B(D_A(M)))) = E_M(D_A(M))$

Even so, Bob goes on with the protocol and sends Mallet a receipt:

$E_M(D_B(E_M(D_A(M))))$

Now, all Mallet has to do is decrypt the message with his private key, encrypt it with Bob's public key, decrypt it again with his private key, and encrypt it with Alice's public key. Voilà! Mallet has M.

It is not unreasonable to imagine that Bob may automatically send Mallet a receipt. This protocol may be embedded in his communications software, for example, and send a receipt automatically upon receipt. It is this willingness to acknowledge the receipt of gibberish that creates the insecurity. If Bob checked the message for comprehensibility before sending a receipt, he could avoid this security problem.

There are enhancements to this attack that allow Mallet to send Bob a different message from the one he eavesdropped on. It is important never to sign arbitrary messages from other people or to decrypt arbitrary messages and give the results to other people.

Foiling the Resend Attack

The above attack works because the encrypting operation is the same as the signature-verifying operation, and the decryption operation is the same as the signature operation. If the protocol used even a slightly different operation for encryption and digital signatures, this would be avoided. Digital signatures with one-way hash functions solve the problem (see Section 2.6); so does using different keys for each operation; so do timestamps, which make the incoming message and the outgoing message different.

In general, then, this protocol is perfectly secure:

(1) Alice signs a message.

(2) Alice encrypts the message and signature with Bob's public key (using a different encryption algorithm than she used for the signature) and sends it to Bob.

(3) Bob decrypts the message with his private key.

(4) Bob verifies Alice's signature.

It is possible to modify the protocol so that Alice encrypts the message before signing it. While this might be suitable for some circumstances, when an intermediate party would need to verify the signature without being able to read the message, in general it is better to encrypt everything. Why give Eve any information at all?

2.8 RANDOM AND PSEUDO-RANDOM SEQUENCE GENERATION

Why even bother with random number generation in a book on cryptography? There's already a random number generator built into most every compiler, a mere function call away. Unfortunately, those random number generators are almost definitely not cryptographically secure, and probably not even very random. Most of them are embarrassingly bad.

Random sequence generators are not random because they don't have to be. Most simple applications, like computer games, need so few random numbers that they hardly notice. However, cryptography is extremely sensitive to the properties of random number generators. Use a poor random sequence generator and you start getting weird correlations and strange results [686,691]. If security depends on your random number generator, weird correlations and strange results are the last things you want.

The problem is that a random number generator doesn't produce a random sequence. It probably doesn't produce anything that looks even remotely like a random sequence. Of course, it is impossible to produce something truly random on a computer. Knuth quotes John von Neumann as saying: "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin" [488]. Computers are deterministic beasts: stuff goes in one end, completely predictable operations occur inside, and different stuff comes out the other end. Put the same stuff in on two separate occasions and the same stuff comes out both times. Put the same stuff into two identical computers, and the same stuff comes out of both of them. There are only a finite number of states in which a computer can exist (a large finite number, but a finite number nonetheless), and the stuff that comes out will always be a deterministic function of the stuff that went in and the computer's current state. That means that any random sequence generator on a computer (at least, on a Turing machine) is, by definition, periodic. Anything that is periodic is, by definition, predictable. And if something is predictable, it can't be random. A true random sequence generator requires some random input; a computer can't provide that.

2.8.1 Pseudo-Random Sequences

The best a computer can produce is a pseudo-random sequence generator. What's that? Many people have taken a stab at defining this formally, but I'll hand-wave here. The sequence's period should be long enough so that a finite sequence of reasonable length, i.e., one that is actually used, is not periodic. That is, if you need a billion random bits, don't choose a sequence generator that repeats after only sixteen thousand bits. These relatively short, non-periodic subsequences should be as indistinguishable as possible from random sequences. For example, they should have about the same number of ones and zeros; about half the runs (sequences of the same bit) should be runs of zeros and the other half should be runs of ones; half the runs should be of length one, one quarter of length two, one eighth of length three, etc. These properties can be empirically measured and then compared to statistical expectations using a chi-square test.

For our purposes, a sequence generator is pseudo-random if it has this property:

1. **It looks random.** This means that it passes all the statistical tests of randomness that we can find. (Start with the ones in [488].)

A lot of effort has gone into producing good pseudo-random sequences on computers. Discussions of generators abound in the academic literature, along with various tests of randomness. All of these generators are periodic (there's no escaping that); but with potential periods of 2^{256} bits and higher, they can be used for the biggest applications.

The problem is still those weird correlations and strange results. Every deterministic generator is going to produce them if you use them in a certain way. And that's what a cryptanalyst will use to attack the system.

2.8.2 Cryptographically Secure Pseudo-Random Sequences

Cryptographic applications demand much more of a pseudo-random sequence generator than do most other applications. Cryptographic randomness doesn't mean just statistical randomness, although that's part of it. For a sequence to be **cryptographically random**, it must have this additional second property:

2. **It is unpredictable.** It must be computationally infeasible to predict what the next random bit will be, given complete knowledge of the algorithm or hardware generating the sequence and all of the previous bits in the stream.

Like any cryptographic algorithm, cryptographically secure pseudo-random sequence generators are subject to attack. Just as it is possible to break an encryption algorithm, it is possible to break a cryptographically secure pseudo-random sequence generator.

2.8.3 Real Random Sequences

Now we're meandering into the domain of philosophers. Is there such a thing as randomness? What is a random sequence? How do you know if a sequence is random? Is "101110100" more random than "101010101"? Quantum mechanics tells us that there is honest-to-goodness randomness in the real world. But is that randomness preserved when brought to the macroscopic world of computer chips and finite-state machines?

Philosophy aside, from our point of view a sequence generator is real-world random if it has this additional third property:

3. **It cannot be reliably reproduced.** If you run the sequence generator twice with the exact same input (at least as exact as humanly possible), you will get two different random sequences.

Additionally, real random sequences cannot be compressed. Cryptographically secure pseudo-random sequences cannot, in practice, be compressed.

The output of a generator with these properties will be good enough for a one-time pad, key generation, and any other cryptographic properties for which one might want it.

CHAPTER 3

Basic Protocols

3.1 KEY EXCHANGE

A common cryptographic technique is to encrypt each individual conversation between two people with a separate key. This is called a session key, because it is used for only one particular communications session. How this common session key gets into the hands of the conversants can be a complicated matter.

Key Exchange with Symmetric Cryptography

(1) Alice calls a Key Distribution Center (KDC) and requests a session key to communicate with Bob.

(2) The KDC generates a random session key. It encrypts two copies of it: one in Alice's key and the other in Bob's key. The KDC also encrypts information about Alice's identity with Bob's key. The KDC sends both copies to Alice.

(3) Alice decrypts her copy of the session key.

(4) Alice sends Bob his copy of the session key and the identity information.

(5) Bob decrypts his copy of the session key and the identity information. He uses the identity information to determine who Alice is. (Presumably Alice knows who Bob is; she called him and only he can decrypt the key.)

(6) Both Alice and Bob use this session key to communicate securely.

This protocol works, but it relies on the absolute security of the KDC. If Mallet corrupts the KDC, the whole network is compromised. He has all of the secret

keys that the KDC shares with each of the users; he can read all past communications traffic and all future communications traffic. All he has to do is to tap the communications lines and listen to the encrypted message traffic.

Key Exchange with Public-Key Cryptography

The hybrid cryptosystem was discussed in Section 2.5.

(1) Bob sends Alice his public key.

(2) Alice generates a random session key, k, encrypts it using Bob's public key, and sends it to Bob.

$E_B(k)$

(3) Bob decrypts Alice's message using his private key to recover the session key.

(4) Both of them encrypt their communications using the same session key.

Key Exchange with Public-Key Cryptography Using a Public-Key Database

In some practical implementations, both Alice's and Bob's signed public keys will be available on a database. This makes the key exchange protocol even easier, and Alice can send a message to Bob even if she has never met him:

(1) Alice gets Bob's public key from a central database.

(2) Alice generates a random session key, encrypts it using Bob's public key, and sends it to Bob.

(3) Bob then decrypts Alice's message using his private key.

(4) Both of them encrypt their communications using the same session key.

Man-in-the-Middle Attack

While Eve cannot do better than try to break the public-key algorithm or attempt a ciphertext-only attack on the ciphertext, Mallet can decrypt messages between Alice and Bob. Mallet is a lot more powerful than Eve. Not only can he listen to messages between Alice and Bob, he can also modify messages, delete messages, and generate totally new ones. Mallet can imitate Bob when talking to Alice and imitate Alice when talking to Bob. Here's how the attack works:

(1) Alice sends Bob her public key. Mallet intercepts this key and sends Bob his own public key.

(2) Bob sends Alice his public key. Mallet intercepts this key and sends Alice his own public key.

(3) When Alice sends a message to Bob, encrypted in "Bob's" public key, Mallet intercepts it. Since the message is really encrypted with his own

DISH-Blue Spike-246
Exhibit 1010, Page 0769

in the database for real signed keys, or he can intercept users' database requests and reply with his phony keys. This enables him to launch a man-in-the-middle attack and read people's communications.

This attack will work, but remember that Mallet has to be a powerful attacker. Intercepting and modifying messages is a lot more difficult than passively sitting on a symmetric network and reading messages as they go by. Active wiretaps are much harder to put in place and much easier to detect. On a broadcast channel such as a radio network, it is almost impossible to replace one message with another.

Key and Message Transmission

There is no reason for Alice and Bob to complete the key-exchange protocol before exchanging messages. In this protocol, Alice sends Bob the message, M, without any previous key-exchange protocol:

(1) Alice generates a random session key, K, and encrypts M using K.

$E_K(M)$

(2) Alice gets Bob's public key from the database.

(3) Alice encrypts K with Bob's public key.

$E_B(K)$

(4) Alice sends both the encrypted message and encrypted session key to Bob.

$E_K(M), E_B(K)$

(For added security against man-in-the-middle attacks, Alice can sign the transmission.)

(1) Bob decrypts Alice's session key, K, using his private key.

(2) Bob decrypts Alice's message using the session key.

This is how public-key cryptography is most often used in a communications system. It can be combined with digital signatures, timestamps, and any other security protocols.

Key and Message Broadcast

There is no reason Alice can't send the encrypted message to several people. In this example, Alice will send the encrypted message to Bob, Carol, and Dave:

(1) Alice generates a random session key, K, and encrypts M using K.

$E_K(M)$

(2) Alice gets Bob's, Carol's, and Dave's public keys from the database.

(3) Alice encrypts K with Bob's public key, encrypts K with Carol's public key, and then encrypts K with Dave's public key.

$E_B(K), E_C(K), E_D(K)$

(4) Alice broadcasts the encrypted message and all the encrypted keys to anybody who cares to receive it.

(5) Only Bob, Carol, and Dave can decrypt the K key using his or her private key.

(6) Only Bob, Carol, and Dave can decrypt Alice's message using K.

This protocol can be implemented on a store-and-forward network. A central server can forward Alice's message to Bob, Carol, and Dave along with their particular encrypted key. The server doesn't have to be secure or trusted, since it will not be able to decrypt any of the messages.

3.2 AUTHENTICATION

When Alice logs into a computer (or an automatic teller, or a telephone banking system, for example), how does the computer know who she is? How does the computer know she is not someone else trying to falsify her identity? Traditionally, passwords solve this problem. Alice enters her password, and the computer confirms that it is correct. Both Alice and the computer know this secret piece of knowledge, and the computer requests it from Alice every time she tries to log in.

What several cryptographers realized is that the computer does not need to know the passwords; the computer just has to be able to differentiate valid passwords from invalid passwords. This is easy with one-way functions [320,621,715,879]. Instead of storing passwords, the computer stores one-way functions of the passwords.

(1) Alice sends the computer her password.

(2) The computer performs a one-way function on the password.

(3) The host compares the result of the one-way function to the value stored in the computer.

Since the computer no longer stores a table of everybody's valid password, the threat of someone breaking into the computer and stealing the password list is mitigated. The list of passwords operated on by the one-way function is useless, because the one-way function cannot be reversed to recover the passwords.

Dictionary Attacks and Salt

Even a file of passwords encrypted with a one-way function is vulnerable. In his spare time, Mallet compiles a list of the 100,000 most common passwords. He operates on all 100,000 of them with the one-way function and stores the results

If each password is about eight bytes, the resulting file will be no more than 800K; it will fit on a single floppy disk. Now, Mallet steals the encrypted password file. He compares the encrypted password with his file of encrypted possible passwords and sees what matches.

This is a **dictionary attack**, and it's surprisingly successful (see Section 7.2.1). Salt is a way to make it more difficult.

Salt is a random string that is concatenated with the password before being operated on by the one-way function. Then, both the salt value and the result of the one-way function are stored in the database. If the number of possible salt values is large enough, this practically eliminates a dictionary attack against commonly used passwords. This is a simple attempt at an **Initialization Vector** (see Section 8.1.3).

A lot of salt is needed. Most UNIX systems use only twelve bits of salt. Even with that, Daniel Klein developed a password-guessing program that cracks 21% of the passwords on a given system in about a week [482]. David Feldmeier and Philip Karn compiled a list of about 732,000 common passwords concatenated with each of 4096 possible salt values. They estimate that 30% of passwords on a given system can be broken with this list [346].

Salt isn't a panacea; increasing the number of salt bits won't solve everything. Salt only protects against general dictionary attacks on a password file, not against a concerted attack on a single password. It obscures people who have the same password on multiple machines, but doesn't make poorly chosen passwords any better.

User Identification with Public-Key Cryptography

Even with salt, the above protocol has serious security problems. First, when Alice types her password into the system, anyone who has access to her data path can read it. She might be accessing her computer through a convoluted transmission path that passes through four industrial competitors, three foreign countries, or two forward-thinking universities, and a partridge in a pear tree. Any one of those points can have an Eve listening to Alice's log-in sequence. Second, anyone with access to the processor memory of the computer system can see the password before the system encrypts it.

Public-key cryptography can solve this problem. The host computer keeps a file of every user's public key; all users keep their own private key. Here is a naive attempt at a protocol. When logging in, the protocol proceeds as follows:

(1) The host sends Alice a random string.

(2) Alice encrypts the string with her private key and sends it back to the host, along with her identity.

(3) The host looks up Alice's public key in its database and decrypts the message using that public key.

(4) If the decrypted string matches what the host sent Alice in the first place, the host allows Alice access to the system.

No one else has access to Alice's private key, so no one else can impersonate the user. More important, Alice never sends her private key over the transmission line to the host. Eve, listening in on the interaction, cannot get any information that would enable her to deduce the private key and impersonate Alice.

✱ The private key is both long and non-mnemonic, and will probably be processed automatically by the user's hardware or communications software. This requires an intelligent terminal that Alice trusts, but neither the host nor the communications path needs to be secure.

In general, it is foolish to encrypt random strings sent by another party; attacks similar to the one discussed in Section 12.4 can be mounted.

✱ In general, secure proof-of-identity protocols take the following form:

(1) Alice performs a computation based on some random numbers and her secret key and sends the result to the host.

(2) The host sends Alice a different random number.

(3) Alice makes a computation based on the random numbers (both the ones she generated and the one she received from the host) and her secret key and sends the result to the host.

(4) The host does a computation on the various numbers received from Alice and her public key to verify that she knows her private key.

(5) If she does, her identity is verified.

If Alice does not trust the host any more than the host does not trust Alice, then Alice will then require the host to prove its identity.

Step (1) might seem unnecessary and confusing, but it is required to prevent attacks against the protocol. Section 12.7 and Section 13.1 mathematically describe several algorithms and protocols. See also [527].

Mutual Authentication Using the Interlock Protocol

Alice and Bob are two users who want to authenticate each other. Each of them has a password: Alice has P_A and Bob has P_B. Here's a protocol that will not work:

(1) Alice and Bob trade public keys.

(2) Alice encrypts P_A with Bob's public key and sends it to him.

(3) Bob encrypts P_B with Alice's public key and sends it to her.

(4) Alice decrypts P_B and verifies that it is correct.

(5) Bob decrypts P_A and verifies that it is correct.

The problem is that Mallet can launch a successful man-in-the-middle attack (see Section 3.1):

(1) Alice and Bob trade public keys. Mallet intercepts both messages. He substitutes his public key for Bob's and sends it to Alice, then he substitutes his public key for Alice's and sends it to Bob.

(2) Alice encrypts P_A with "Bob's" public key and sends it to him. Mallet intercepts the message, decrypts P_A with his private key, re-encrypts it with Bob's public key and sends it on to him.

(3) Bob encrypts P_B with Alice's public key and sends it to her. Mallet intercepts the message, decrypts P_B with his private key, re-encrypts it with Alice's public key, and sends it on to her.

(4) Alice decrypts P_B and verifies that it is correct.

(5) Bob decrypts P_A and verifies that it is correct.

From what Alice and Bob see, nothing is different. However, Mallet knows both P_A and P_B.

Davies and Price describe how the interlock protocol can defeat this attack [249]:

(1) Alice and Bob trade public keys.

(2) Alice encrypts P_A with Bob's public key and sends half of it to him.

(3) Bob encrypts P_B with Alice's public key and sends half of it to her.

(4) Alice sends the other half of encrypted P_A to Bob.

(5) Bob combines the two halves, decrypts P_A, and verifies that it is correct.

(6) Bob sends the other half of encrypted P_B to Alice.

(7) Alice combines the two halves, decrypts P_B, and verifies that it is correct.

Steve Bellovin and Michael Merritt discuss ways to attack this protocol [60]. If Alice is a user and Bob is a host, Mallet can pretend to be Bob, complete steps (1) through (5) of the protocol with Alice, and then drop the connection. True artistry demands Mallet do this by simulating line noise or network failure, but the final result is that Mallet has Alice's password. He can then connect with Bob and complete the protocol. Voilà. Mallet now has Bob's password.

The protocol can be modified so that Bob gives his password before Alice, under the assumption that the user's password is much more sensitive than the host's password. This falls to a more complicated attack, also described in [60].

SKID

SKID2 and SKID3 are secret-key identification protocols developed for RACE's RIPE project [727] (see Section 18.8). They use a keyed one-way hash function (a MAC) to provide security, and both assume that both Alice and Bob share a secret key, K.

SKID2 allows Bob to prove his identity to Alice. Here's the protocol:

(1) Alice chooses a random number, R_A. (The RIPE document specifies a 64-bit number.) She sends it to Bob.

(2) Bob chooses a random number, R_B. (The RIPE document specifies a 64-bit number.) He sends Alice:

$R_B, H_K(R_A, R_B, Bob)$

H_K is the MAC. (The RIPE document suggests the RIPE-MAC function—see Section 14.14.) Bob is Bob's name.

(3) Alice computes $H_K(R_A, R_B, Bob)$ and compares it with what she received from Bob. If the results are identical, then Alice knows that she is communicating with Bob.

SKID3 provides mutual authentication between Alice and Bob. Steps (1) through (3) are identical to SKID2, and then the protocol proceeds with:

(4) Alice sends Bob:

$H_K(R_B, Alice)$

Alice is Alice's name.

(5) Bob computes $H_K(R_B, Alice)$, and compares it with what he received from Bob. If the results are identical, then Bob knows that he is communicating with Alice.

3.3 AUTHENTICATION AND KEY EXCHANGE

These protocols solve a general computer problem: Alice and Bob are sitting on other ends of the network. They want to talk securely. How can Alice and Bob exchange a secret key and at the same time each be sure they are talking to the other and not to Mallet?

Wide-Mouth Frog

The Wide-Mouth Frog protocol [180] is probably the simplest symmetric key-management protocol that uses a trusted server. Both Alice and Bob share a secret key with Trent. These keys are themselves distributed secretly and authentically over some external channel, which we shall assume to be secure. The keys are just used for key distribution and not to encrypt any actual messages between users. Just by using two messages, a session key is transferred from Alice to Bob:

(1) Alice concatenates a timestamp, T_A, Bob's name, B, and a random session key, K, and encrypts the whole message with the key she shares with Trent. She sends this to Trent, along with her identifier, A.

$A, E_A(T_A, B, K)$

(2) Trent concatenates a timestamp, T_B, Alice's name, and the random session key and encrypts the whole message with the key he shares with Bob. Trent sends it to Bob:

$E_B(T_B,A,K)$

The biggest assumption made in this protocol is that Alice is competent enough to generate good session keys. Remember that random numbers aren't easy to generate; it might be more than Alice can be trusted to do properly.

Yahalom

This protocol is by Yahalom [180]. Like the previous protocol, both Alice and Bob share a secret key with Trent.

(1) Alice concatenates her name and a random number, R_A, and sends it to Bob.

A,R_A

(2) Bob concatenates Alice's name, Alice's random number, another random number, R_B, and encrypts it with the key he shares with Trent. He sends this to Trent, along with his name.

$B,E_B(A,R_A,R_B)$

(3) Trent generates two messages. The first consists of Bob's name, a random session key for Alice and Bob, K, Alice's random number, and Bob's random number, all encrypted with the key he shares with Alice. The second consists of Alice's name and the random session key, encrypted with the key he shares with Bob. He sends both messages to Alice.

$E_A(B,K,R_A,R_B),E_B(A,K)$

(4) Alice decrypts the message encrypted with her key, extracts K, and confirms that R_A has the same value as it did in step (1). Alice sends Bob two messages. The first is the message received from Trent, encrypted with Bob's key. The second is R_B, encrypted with the session key.

$E_B(A,K),E_K(R_B)$

(5) Bob decrypts the message encrypted with his key, extracts K, and confirms that R_B has the same value as it did in step (2).

At the end, Alice and Bob are each convinced that they are talking to the other and not to a third party. The novelty here is that Bob is the first one to contact Trent, who only sends one message to Alice.

Needham and Schroeder

This protocol, invented by Needham and Schroeder [645], also uses symmetric cryptography and Trent.

(1) Alice sends a message to Trent consisting of her name, A, Bob's name, B, and some random value R_A.

A,B,R_A

(2) Trent generates a random session key, K. He encrypts a message consisting of K and Alice's name with the secret key he shares with Bob. Then he encrypts Alice's random value, Bob's name, the key, and the encrypted message with the secret key he shares with Alice. Finally, he sends her the encrypted message:

$E_A(R_A,B,K,E_B(K,A))$

(3) Alice decrypts the message and extracts K. She confirms that R_A is the same value that she sent Trent in step (1). Then she sends Bob the message that Trent encrypted in his key.

$E_B(K,A)$

(4) Bob decrypts the message and extracts K. He then generates another random value, R_B. He encrypts the message with K and sends it to Alice.

$E_K(R_B)$

(5) Alice decrypts the message with K. She generates R_B-1 and encrypts it with K. Then she sends the message back to Bob.

$E_K(R_B-1)$

(6) Bob decrypts the message with K and verifies that it is R_B-1.

All of this fussing around with R_A and R_B and R_B-1 is to guarantee that there are no replay attacks. The presence of R_A in step (3) assures Alice that Trent's message is legitimate and not a replay of an old response from a previous execution of the protocol. When Alice successfully decrypts R_B and sends Bob R_B-1 in step (5), Bob is ensured that Alice's messages are not replays from earlier executions of the protocol.

The biggest security problem with this protocol is that old session keys are valuable. If Mallet gets access to an old K, he can launch a successful attack [272]. All he has to do is record Alice's messages to Bob in step (3). Then, once he has K, he can pretend to be Alice:

(1) Mallet sends Bob the following message:

$E_B(K,A)$

(2) Bob extracts K, generates R_B, and sends "Alice":

$E_K(R_B)$

(3) Mallet intercepts the message, decrypts it with K, and sends Bob:

$E_K(R_B-1)$

(4) Bob verifies that "Alice's" message is R_B-1.

Now, Mallet has Bob convinced that he is Alice.

A stronger protocol, using timestamps, can defeat this attack [272]. This protocol requires a secure and accurate system clock, not a trivial problem in itself. This modified protocol is the basis for the Kerberos authentication protocols (see Section 17.4).

There are even more drastic consequences if K_A is ever compromised. Mallet can use it to obtain session keys to talk with Bob (or anyone else he wishes to talk to). Even worse, Mallet can continue to do this even after Alice changes her key [47].

Needham and Schroeder attempted to correct these problems in a modified version of their protocol [646]. Their new protocol is essentially the same as the Otway-Rees protocol, published in the same issue of the same journal.

Otway-Rees

Otway and Rees's protocol also uses symmetric cryptography [683]. As usual, Trent shares a separate key with everyone on the network.

(1) Alice generates a message consisting of an index number, I, her name, A, Bob's name, B, and a random number, R_A, all encrypted in the key she shares with Trent. She sends this message to Bob along with the index number, her name, and his name:

$I,A,B,E_A(R_A,I,A,B)$

(2) Bob generates a message consisting of a new random number, R_B, the index number, Alice's name, and Bob's name, all encrypted in the key he shares with Trent. He sends it to Trent, along with Alice's encrypted message, the index number, her name, and his name:

$I,A,B,E_A(R_A,I,A,B),E_B(R_B,I,A,B)$

(3) Trent generates a random session key, K. Then he creates two messages. One is Alice's random number and the session key, encrypted in the key he shares with Alice. The other is Bob's random number and the session key, encrypted in the key he shares with Bob. He sends these two messages, along with the index number, to Bob:

$I,E_A(R_A,K),E_B(R_B,K)$

(4) Bob sends Alice the message encrypted in her key, along with the index number:

$I,E_A(R_A,K)$

Assuming that all the random numbers match, and the index number hasn't changed along the way, Alice and Bob are now convinced of each other's identity, and they have a secret key with which to communicate.

Kerberos

Kerberos has been implemented and is discussed in detail in Section 17.4. The basic Kerberos Version 5 protocol is as follows: Alice and Bob each share keys with Trent. Alice wants to generate a session key for a conversation with Bob.

(1) Alice sends a message to Trent with her identity, A, and Bob's identity, B.

A,B

(2) Trent generates a message with a timestamp, T, a lifetime, L, a random session key, K, and Alice's identity. He encrypts this in the key he shares with Bob. Then he takes the timestamp, the lifetime, the session key, and Bob's identity, and encrypts this in the key he shares with Alice:

$E_A(T,L,K,B),E_B(T,L,K,A)$

(3) Alice generates a message with her identity and the timestamp, encrypts it in K, and sends it to Bob. Alice also sends Bob the message encrypted in Bob's key from Trent.

$E_K(A,T),E_B(T,L,K,A)$

(4) Bob sends a message consisting of the timestamp plus one, encrypts it in K, and sends it to Alice.

$E_K(T+1)$

This protocol works, but it assumes that everyone's clocks are synchronized with Trent's clock. In practice, the effect is obtained by synchronizing clocks to within a few minutes of a secure time server and detecting replays within the time interval.

SPX

The SPX protocols, developed at Digital Equipment Corporation, also provide for mutual authentication and key exchange [851,850]. Unlike the other protocols, SPX uses both public-key and symmetric cryptography. Alice and Bob each has a private key. Trent has signed copies of their public keys.

(1) Alice sends a message to Trent, consisting of Bob's identity, B.

B

(2) Trent sends Alice Bob's public key, K_B, signed in Trent's private key, T.

$S_T(K_B)$

(3) Alice verifies Trent's signature to confirm that the key she received is actually Bob's public key. She generates a random secret key, K, and a random public-key/private-key key pair, K_P. She encrypts the time, T_A, with K. Then she signs a key lifetime, L, her identification, A, and K_P with

her private key, K_A. Finally, she encrypts K with Bob's public key, and signs it with K_P. She sends all of this to Bob.

$E_K(T_A), S_{K_T}(L,A,K_P), S_{K_P}(K_B(K))$

(4) Bob sends a message to Trent (this may be a different Trent), consisting of Alice's identity.

A

(5) Trent sends Bob Alice's public key, signed in Trent's private key.

$S_T(K_A)$

(6) Bob verifies Trent's signature to confirm that the key he received is actually Alice's public key. He then verifies Alice's signature and recovers K_P. He verifies the signature and uses his private key to recover K. Then he decrypts T_A to make sure this is a current message.

(7) If mutual authentication is required, Bob encrypts a new timestamp, T_B, with K, and sends it to Alice.

$E_K(T_B)$

(8) Alice decrypts T_B with K to make sure that the message is current.

DEC has a working implementation of the SPX protocols. Additional information can be found in [18].

Other Protocols

There are many other protocols in the literature. The CCITT X.509 protocols are discussed in Section 17.6. KryptoKnight is discussed in Section 17.5. Another protocol, Encrypted Key Exchange, is discussed in Section 16.2.

3.4 MULTIPLE-KEY PUBLIC-KEY CRYPTOGRAPHY

In public-key cryptography, there are two keys. A message encrypted with one key can be decrypted with the other. Usually one key is private and the other is public. However, let's assume that Alice has one key and Bob has the other. Now Alice can encrypt a message so that only Bob can decrypt it, and Bob can encrypt a message so that only Alice can read it.

This concept was generalized by Colin Boyd [134]. Imagine a variant of public-key cryptography with three keys: K_A, K_B, and K_C. Alice has the first, Bob the second, and Carol the third. In addition, Dave has both K_A and K_B. Ellen has both K_B and K_C. Frank has both K_C and K_A. Any number of keys can be used to encrypt a message. The remaining keys are required to decrypt that message.

Alice can encrypt a message with K_A so that Ellen, with K_B and K_C, can decrypt it, as can Bob and Carol in collusion. Bob can encrypt a message so that Frank can read it, and Carol can encrypt a message so that Dave can read it. Dave can encrypt

a message with K_A so that Ellen can read it, with K_B so that Frank can read it, or with both K_A and K_B so that Carol can read it. Similarly, Ellen can encrypt a message so that either Alice, Dave, or Frank can read it. No other pairs can communicate. This is summarized in Table 3.1:

TABLE 3.1
Three-Key Message Encryption

ENCRYPTED WITH KEYS	MUST BE DECRYPTED WITH KEYS
K_A	K_B and K_C
K_B	K_A and K_C
K_C	K_A and K_B
K_A and K_B	K_C
K_A and K_C	K_B
K_B and K_C	K_A

This can be extended to n keys. If a certain subset of the keys is used to encrypt the message, then the other keys are required to decrypt the message.

Broadcasting a Message

Imagine that you have three operatives out in the field: Alice, Bob, and Carol. You want to be able to send messages to subsets of them but don't know which subsets in advance. You can either encrypt the message separately for each person or give out keys for every possible combination. The first option requires a lot more communications traffic; the second requires a lot more keys.

Multiple-key cryptography is much easier. You give Alice K_A and K_B, Bob K_B and K_C, and Carol K_C and K_A. Now you can talk to any subset you want. If you want to send a message so that only Alice can read it, encrypt it with K_C. When Alice receives the message, she decrypts it with K_A and then K_B. If you want to send a message so that only Bob can read it, encrypt it with K_A; so that only Carol can read it, with K_B. If you want to send a message so that both Alice and Bob can read it, encrypt it with K_A and K_C, and so on.

This might not seem exciting, but with 100 operatives one can appreciate the ease of this scheme. If you want to send messages to subsets of those operatives, you have three choices: you can share a key with each operative (100 keys total) and send individual messages. You can distribute $2^{100}-1$ keys to account for every possible subset. Or, you can use this scheme; it works with only one encrypted message and 100 different keys.

There are other techniques for message broadcasting. These are discussed in Section 16.4.

CHAPTER 4

Intermediate Protocols

4.1 SUBLIMINAL CHANNEL

Alice and Bob have been arrested and are going to jail. He's going to the male prison, and she's going to the female prison. Their only means of communication will be via messages. Walter, the warden, is willing to let Alice and Bob exchange messages, but he won't allow them to be encrypted. Walter expects that they are going to coordinate an escape plan, so he wants to be able to read everything.

Walter also hopes that he can deceive either Alice or Bob. He wants one of them to accept a fraudulent message as a genuine message from the other. Alice and Bob are willing to go along with this risk of deception, otherwise they cannot communicate at all, and they have to coordinate their plans. To do this they have to deceive the warden and find a way of communicating secretly. They have to set up a subliminal channel, a covert communications channel between them, in full view of Walter, even though the messages themselves contain no secret information. Through the exchange of perfectly innocuous, signed messages they will pass secret information back and forth and fool Walter, even though Walter is watching all the communications.

An easy subliminal channel might be the number of words in a sentence. An odd number of words in a sentence might correspond to "1," while an even number of words might correspond to "0." So, while you read this seemingly innocuous paragraph, I have sent my operatives in the field the message "101." The problem with that algorithm is there is no key; the security is dependent on the secrecy of the algorithm. Better security is certainly possible.

Peter Wayner's mimic functions obfuscate messages. These functions hide the identity of a message by modifying it so that its statistical profile resembles that of something else: the classifieds section of the *New York Times*, a play by

relevant for content marking

Shakespeare, or a newsgroup on the Internet [871,872]. There are no keys involved; this is a restricted algorithm.

Gustavus Simmons invented the concept of a subliminal channel in a conventional digital signature algorithm [821]. Since the subliminal messages are hidden in what looks like normal digital signatures, this is a form of obfuscation. Walter sees signed innocuous messages pass back and forth, but he completely misses the information being sent over the subliminal channel. In fact, the subliminal channel signature algorithm is indistinguishable from a normal signature algorithm, at least to Walter. Walter not only cannot read the subliminal message, but he also has no idea that one is even present. (Of course, any warden who gives his prisoners computers and high-speed modems deserves what he gets.)

In general the protocol looks like this:

(1) Alice generates an innocuous message, at random.

(2) Using a secret key shared with Bob, Alice signs the innocuous message in such a way as to hide her subliminal message in the signature. (This is the meat of the subliminal channel protocol; see Section 16.6.)

(3) Alice sends this signed message to Bob via Walter.

(4) Walter reads the innocuous message and checks the signature. Finding nothing amiss, he passes the signed message to Bob.

(5) Bob checks the signature on the innocuous message, confirming that the message came from Alice.

(6) Bob ignores the innocuous message and, using the secret key he shares with Alice, extracts the subliminal message.

What about cheating? Walter doesn't trust anyone and no one trusts him. He can always prevent communication, but he has no way of introducing phony messages. Since he can't generate any valid signatures, Bob will detect his attempt in step (3). And since he does not know the shared key, he can't read the subliminal messages. Even more important, he has no idea that the subliminal messages are there. Signed messages using a digital signature algorithm look no different from signed messages with subliminal messages embedded in the signature.

Cheating between Alice and Bob is more problematic. In some implementations of a subliminal channel, the secret information Bob needs to read the subliminal message is the same information Alice needs to sign the innocuous message. If this is the case, Bob can impersonate Alice. He can sign messages purporting to come from her, and there is nothing Alice can do about it. If she is to send him subliminal messages, she has to trust him not to abuse her private key.

Other subliminal channel implementations don't have this problem. A secret key shared by Alice and Bob allows Alice to send Bob subliminal messages, but it is not the same as Alice's private key and does not allow Bob to sign messages. Alice does not have to trust Bob not to abuse her private key.

Applications of Subliminal Channel

The most obvious application of the subliminal channel is in a spy network. If everyone is sending and receiving signed messages, spies will not be noticed sending subliminal messages in signed documents. Of course, the enemy's spies can do the same thing.

Using a subliminal channel, Alice could safely sign a document under threat. She would, when signing the document, embed the subliminal message, saying "I am being coerced." Other applications are more subtle. A company can sign documents and embed subliminal messages, allowing them to be tracked throughout the document's lifespan. The government can "mark" digital currency. A malicious signature program can leak the private key. The possibilities are endless.

Subliminal-Free Signatures

Alice and Bob are sending signed messages to each other, negotiating the terms of a contract. They use a digital signature protocol. However, this contract negotiation has been set up as a cover for Alice's and Bob's spying activities. When they use the digital signature algorithm, they don't care about the messages they are signing. They are using a subliminal channel in the signatures to send secret information to each other. The counterespionage service, however, doesn't know that the contract negotiations and the use of signed messages are just cover-ups.

The use of subliminal channels has led people to create subliminal-free signature schemes. These are digital signature schemes that cannot be modified to contain a subliminal channel. See [283,284].

4.2 UNDENIABLE DIGITAL SIGNATURES

The Alice Software Company distributes DEW (Do-Everything-Word™). To ensure that their software is virus-free, they include a digital signature. However, they want only legitimate buyers of the software, not pirates, to be able to verify the signature. At the same time, if copies of DEW are found containing a virus, there should be no way for the Alice Software Company to deny a valid signature.

Conventional digital signatures can be copied exactly. Sometimes this property is useful, as in the dissemination of public announcements. Other times they could be a problem. Imagine a digitally signed personal or business letter. If many copies of that document were floating around, each of which could be verified by anyone, this could lead to embarrassment or blackmail. The best solution is a digital signature that can be proven to be valid, but one that the recipient cannot show to a third party without the signer's consent.

Undeniable signatures, invented by David Chaum [207], are suited to these tasks. Like a normal digital signature, an undeniable signature depends on the signed document and the signer's private key. But unlike normal digital signatures, an undeniable signature cannot be verified without the signer's consent.

The mathematics behind this protocol can be found in Section 16.7, but the basic idea is simple:

(1) Alice presents Bob with a signature.

(2) Bob generates a random number and sends it to Alice.

(3) Alice does a calculation, using the random number and her private key, and sends Bob the result. Alice could only do this calculation if the signature is valid.

(4) Bob confirms this.

Bob can't turn around and convince Carol that Alice's signature is valid, because Carol doesn't know that Bob's numbers are random. He could have easily worked the protocol backwards on paper, without any help from Alice, and then shown Carol the result. Carol cannot be convinced that Alice's signature is valid unless she does the protocol with Alice herself. This might not make much sense now, but it will once the mathematics are shown.

This solution isn't perfect. Yvo Desmedt and Moti Yung show that it is possible, in some applications, for Bob to convince Carol that Alice's signature is valid [292].

For instance, Bob buys a legal copy of DEW. He can validate the signature on the software package whenever he wants. Then, Bob convinces Carol that he's a salesman from the Alice Software Company. He sells her a pirated copy of DEW. When Carol tries to validate the signature with Bob, he simultaneously validates the signature with Alice. When Carol sends him the random number, he then sends it to Alice. When Alice replies, he sends it to Carol. Carol is convinced that she is a legitimate buyer of the software, even though she isn't. This attack is an instance of the chess grandmaster problem and is discussed in detail in Section 5.4.

4.3 FAIL-STOP DIGITAL SIGNATURES

Let's say Eve is a very powerful adversary. She has vast computer networks, rooms full of Cray computers: orders of magnitude more computing power than Alice. All of these computers function day and night, trying to break Alice's private key. Then, finally—success. Eve can now impersonate Alice, forging her signature on documents at will.

Fail-stop digital signatures, introduced by Birgit Pfitzmann and Michael Waidner [693], prevent this kind of cheating. If Eve forges signatures in this manner, then Alice can prove they are forgeries. If Alice signs a document and then disavows the signature, claiming forgery, a court can verify that it is not a forgery.

The basic idea behind fail-stop signatures is that for every possible public key, there are many possible private keys that would work with it. Each of these private keys yields many different possible signatures. However, Alice has only one private key and can compute just one signature. Alice doesn't know any of the other private keys.

Eve is trying to break Alice's private key. (In this case, Eve could be Alice, trying to compute a second private key for herself.) She collects signed messages and, using her array of Cray computers, tries to recover her private key. Even if she encryption key but also the exact nature of the transformation itself is unknown to a potential attacker.

* MDC is unpatented. The source code is unlicensed. Anyone can use it at any time, in any way, royalty-free [406].

11.14 OTHER BLOCK ALGORITHMS

Four Japanese cryptographers presented an algorithm based on chaos theory at EUROCRYPT '91 [413,414]. Biham cryptanalyzed the algorithm at the same conference [98].

There are many more block algorithms available outside the cryptology community. Some are in use by various government and military organizations. I have no information about any of those. There are also dozens of proprietary commercial algorithms. Some might be good; most are probably not. If companies do not feel that their interests are served by making their algorithms public, it is best to assume they're right and avoid the algorithm.

11.15 WHICH BLOCK ALGORITHM IS BEST?

That's a tough question. DES is still secure, unless you need your secrets to remain secret for many years or fear a major government. If you need security that lasts decades or fear the cryptanalytic efforts of major governments, use triple DES with three independent keys.

The other algorithms aren't worthless. Assuming that there are no cryptanalytic results that are being kept secret (certainly a possibility), REDOC-II, REDOC-III, and Khufu are still secure. Even so, these algorithms have not been analyzed by enough people to make me feel safe. Khufu, which requires precomputation of the S-boxes based on the key, may not be suitable for some on-the-fly applications. The one-way hash-function-based algorithms have promise; Luby-Rackoff and MDC are as secure as their underlying one-way hash functions.

My favorite algorithm is IDEA. Its 128-bit key, combined with its resistance to any public means of cryptanalysis, gives me a warm fuzzy feeling about the algorithm. I wish it had a 128-bit block, but you can't have everything. The algorithm is new, but it is be analyzed by a lot of different groups. There could be some devastating cryptanalytic news tomorrow; today I'm betting on IDEA.

CHAPTER 12

Public-Key Algorithms

12.1 BACKGROUND

The concept of public-key cryptography was invented by Whitfield Diffie and Martin Hellman, and independently by Ralph Merkle. This contribution to the field was the notion that keys could come in pairs—an encryption key and a decryption key—and that it could be unfeasible to generate one key from the other. Diffie and Hellman first presented this concept at the 1976 National Computer Conference [298]; a few months later, their seminal paper, "New Directions in Cryptography," appeared in *IEEE Transactions on Information Theory* [299]. (Due to the glacial publishing speed of *Communications of the ACM*, Merkle's first contribution to the field didn't appear until 1978 [586].)

The first public-key algorithms became public at the same time that DES was being discussed as a proposed standard. This resulted in some partisan politics in the cryptographic community. As Diffie described it [297]:

> The excitement public key cryptosystems provoked in the popular and scientific press was not matched by corresponding acceptance in the cryptographic establishment, however. In the same year that public key cryptography was discovered, the National Security Agency (NSA), proposed a conventional cryptographic system, designed by International Business Machines (IBM), as a federal *Data Encryption Standard* DES. Marty Hellman and I criticized the proposal on the ground that its key was too small, but manufacturers were gearing up to support the proposed standard and our criticism was seen by many as an attempt to disrupt the standards-making process to the advantage of our own work. Public key cryptography in its turn was attacked, in sales literature [624] and technical

papers [483,643] alike, more as though it were a competing product than a recent research discovery. This, however, did not deter the NSA from claiming its share of the credit. Its director, in the words of the *Encyclopaedia Britannica* [824], "pointed out that two-key cryptography had been discovered at the agency a decade earlier," although no evidence for this claim was ever offered publicly.

Since 1976, numerous public-key cryptography algorithms have been proposed. Many of these are insecure. Of the ones that are still considered secure, many are impractical. Either they have an impractically large key, or the ciphertext is much larger than the plaintext.

Only a few algorithms are both secure and practical. These algorithms are generally based on one of the hard problems discussed in Section 9.2. While it is theoretically possible that someone may come up with a fast solution, however, not many experts think it is likely.

Of these secure and practical public-key algorithms, some are only suitable for key distribution. Others are suitable for encryption (and by extension for key distribution). Others are only suitable for digital signatures. Only two algorithms are suitable for both encryption and digital signatures: RSA and ElGamal. All of these algorithms are slow. The encryption and decryption rate is much slower than with symmetric algorithms; usually it's too slow to support bulk data encryption.

Security of Public-Key Algorithms

Since cryptanalysts have access to the public key, they can always choose any message to encrypt. This means that cryptanalysts, given $C = E_K(P)$, can guess the value of P and easily check their guess. This is a serious problem if the number of possible plaintext messages is small enough to allow exhaustive search, but can be solved by padding messages with a string of random bits. This means that identical plaintext messages will encrypt to different ciphertext messages. (For more about this concept, see Section 16.16.)

*

Public-key algorithms are designed to resist chosen-plaintext attacks; their security is based both on the difficulty of deducing the secret key from the public key and the difficulty of deducing the plaintext from the ciphertext. However, most public-key algorithms are particularly susceptible to what is called a chosen-ciphertext attack:

> A chosen-ciphertext attack is when cryptanalysts choose messages and have access to the decryption of those messages with the private key. Their job is to deduce the key or an algorithm to decrypt any new messages encrypted with the same key.
>
> *Given:* $C_1, P_1 = D_k(C_1), C_2, P_2 = D_k(C_2), \ldots C_i, P_i = D_k(C_i)$, where cryptanalysts choose $C_1, C_2, \ldots C_i$
>
> *Deduce:* K, or an algorithm to infer P_{i+1} from $C_{i+1} = E_k(P_{i+1})$

It is important to understand the significance of this attack. Even if we have a public-key cryptosystem that is provably secure and use a protocol that looks safe

cryptanalysts may still recover plaintext messages. The inherent problem is a direct result of the most useful characteristic of public-key cryptography: everyone can use the public key.

Consequently, it is important to look at the system as a whole and not just at the individual parts. Good public-key protocols are designed so that the various parties don't decrypt arbitrary messages generated by other parties—the proof-of-identity protocols are a good example (see Section 5.4).

12.2 DIFFIE-HELLMAN

Diffie-Hellman is the first public-key algorithm invented [299]. It gets its security from the difficulty of calculating discrete logarithms in a finite field, as compared with the ease of calculating exponentiation in the same field. Diffie-Hellman can be used for key distribution, but it cannot be used to encrypt and decrypt messages. Alice and Bob can use this algorithm to generate a secret key.

The math is simple. First, Alice and Bob agree on two large integers, n and g, such that g is less than n but greater than 1. These two integers don't have to be secret; Alice and Bob can agree to them over some insecure channel. They can even be common among a group of users. It doesn't matter.

Then, the protocol goes as follows:

(1) Alice chooses a random large integer x and computes

$$X = g^x \bmod n$$

(2) Bob chooses a random large integer y and computes

$$Y = g^y \bmod n$$

(3) Alice sends X to Bob, and Bob sends Y to Alice. (Note that Alice keeps x secret, and Bob keeps y secret.)

(4) Alice computes $k = Y^x \bmod n$.

(5) Bob computes $k' = X^y \bmod n$.

Both k and k' are equal to $g^{xy} \bmod n$. No one listening on the channel can compute that value; they only know n, g, X, and Y. Unless they can compute the discrete logarithm and recover x or y, they do not solve the problem. So, k is the secret key that both Alice and Bob computed independently.

The choice of g and n can have a substantial impact on the security of this system. The modulus n should be a prime; more importantly $(n-1)/2$ should also be a prime [703]. And g should be a primitive root mod n. And most important, n should be large: at least 512 bits long. Using 1028 bits would be better.

Diffie-Hellman Extended

This algorithm also works in finite fields [703]. Shmuley and Kevin McCurley studied a variant of the algorithm where the modulus is a composite number [812,578]. V. S. Miller and Neal Koblitz extended this algorithm to elliptic curves

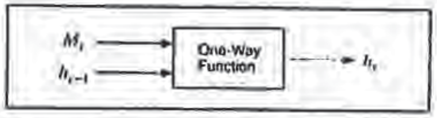


Figure 14.1
One-way hash function

Typically, the information hashed should contain some kind of binary representation of the length of the entire message. This technique overcomes a potential security problem resulting from messages with different lengths possibly hashing to the same value [587,236].

Various researchers have theorized that if the one-way function is secure for a single block, then this method of hashing an arbitrary-length string of blocks is also secure [632,589,236]. Proven results are more limited.

14.2 SNEFRU

Snefru is a one-way hash function designed by Ralph Merkle [589]. Snefru hashes arbitrary-length messages into either 128-bit or 256-bit values.

First the message is broken into chunks, each $512-m$ in length. (The variable m is the length of the hash value.) If the output is a 128-bit hash value, then the chunks are each 384 bits long; if the output is a 256-bit hash value, then the chunks are each 256 bits long.

The heart of the algorithm is a function H, which hashes a 512-bit value into an m-bit value. It was designed so that it was easy to compute the hash of an input but computationally unfeasible to compute an input that generates a specific hash value.

The first m bits of the H's output are the hash of the block; the rest are discarded. The next block is appended to the hash of the previous block and then hashed again. (The initial block is appended to a string of *zeros*.) After the last block (if the message isn't an even number of blocks long, *zeros* are used to pad the last block), the first m bits are appended to a binary representation of the length of the message and hashed one final time.

Function H is based on another function, E, which is a reversible block cipher function that operates on 512-bit blocks. H is the last m bits of the output of E XORed with the first m bits of the input of E.

The security of Snefru resides in function E, which randomizes data in several passes. Each pass is composed of 64 randomizing rounds. In each round a different byte of the data is used as an input to an S-box; the output word of the S-box is XORed with two neighboring words of the message. The S-boxes are constructed in a manner similar to those in Khafre. There are also some rotations thrown in. Originally Snefru was designed with two passes.

Cryptanalysis of Snefru

Using differential cryptanalysis, Biham and Shamir demonstrated the insecurity of two-pass Snefru (128-bit hash value) by showing that the following messages hashed to the same value [106]:


```
3FE15E26 2307C030 C7089999 90EFC48F A04087EE 16493302
00046085 00003415 00000000 00000000 00000000 00000000

3FE15F26 2307C030 C7089999 90CFC48F A040B7E1 D74AF7AE
096C7885 C19EF029 00000000 00000000 00000000 00000000

Common hash value: C8FF5E2C 8F9CF7C7 E000DAA7 E4F9044E
```

Also, that the following four messages hashed to the same value:

```
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000

00000000 F1301600 130FC53E 4CC38093 37461661 CCD80940
2409D35F 71471FDE 00000000 00000000 00000000 00000000

00000000 1D197F00 2ABD3F6F CF33F301 8674966A 816E5051
ACD9A905 53C1D180 00000000 00000000 00000000 00000000

00000000 E98C8300 1F777A47 85271F34 A0497488 44CC8867
8E4B0EFC 18131756 00000000 00000000 00000000 00000000

Common hash value: 2E88E244 E9D4A20B B2002F0B 7200E1E6
```

On a personal computer, their attack finds pairs of messages that hash to the same value within three minutes and a message that hashes to a given hash value in about an hour.

On 128-bit Snefru, their attacks work better than brute force for four passes or less. A birthday attack against Snefru takes 2^{64} operations; differential cryptanalysis can find a pair of messages that hash to the same value in $2^{28.5}$ operations for three-pass Snefru and $2^{44.5}$ operations for four-pass Snefru. Finding a message that hashes to a given value by brute force requires 2^{128} operations; differential cryptanalysis takes 2^{56} operations for three-pass Snefru and 2^{88} operations for four-pass Snefru.

The results for Snefru with longer hash lengths were also better than brute force. Although Biham and Shamir didn't analyze 256-bit hash values, they extended their analysis to 224-bit hash values. Compared to a birthday attack, which requires 2^{112} operations, they can find messages that hashed to the same value in $2^{12.5}$ operations for two-pass Snefru, 2^{33} operations for three-pass Snefru, and 2^{88} operations for four-pass Snefru.

Currently, Merkle recommends using Snefru with at least eight passes [592]. However, with this many passes the algorithm is significantly slower than either MD5 or SHA.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☑ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

Chapter 10 introduces the digital audio tape (DAT) format, a rotary head system used to record two audio channels, along with extensive subcode. DAT design is examined, including azimuth recording, track format, modulation, and error correction. The prerecorded DAT format is explained. Professional DAT applications of timecode recording and editing are presented, along with a look at the S-DAT format.

Chapter 11 discusses optical storage and transmission. The chapter begins with a review of optical phenomena such as diffraction, resolution, and polarization. Optical storage systems are discussed, including both nonerasable and erasable disk systems. Each system may be implemented with a variety of technologies, which are explained. The chapter concludes with a look at fiber optics. The advantages of fiber optics, the operation of such a system, and the limitations of fiber optic interconnection are discussed.

Chapter 12 is devoted to the compact disc, the optical, random access, digital audio disc system designed to replace the LP as the dominant consumer playback medium. The physical characteristics of discs and the nature of the data encoded on discs are presented. The theory of operation of the player is explained, following the signal path of the data from the disc to the player's output. The laser pickup, EFM, CIRC error correction, and other topics are also discussed, as are alternative CD formats such as CD-ROM, CD-V, CD-I, CD-WO, and CD+G.

Chapter 13 tackles the topic of digital signal processing. Linearity, time invariance, complex numbers, impulse response, convolution, and transforms are explained in a largely nonmathematical fashion. Digital filter theory is discussed, with a look at both FIR and IIR filters. Parameters for filter design are presented, and the circuits used for digital effects such as delay and reverberation are explained. The chapter concludes with a look at DSP chips and their place in a digital system.

Chapter 14 looks forward to the widespread use of digital audio workstations, following the introduction of digital mixing consoles. Already, workstations have changed post-production methods in many professional audio applications. The flexibility and complexity of workstations may even encourage the use of artificial intelligence in the audio profession. In particular, expert systems may find an important role in the studio. The chapter discusses this emerging technology and presents specific examples of workstation operation.

Much of the material in this book stems from the work of the many pioneers and leaders in the field of digital audio technology. For their efforts in developing the potential of this young science, we all owe them a tremendous debt.

189

1 Audio Basics

Digital audio is a highly sophisticated technology. It extends the frontiers of engineering and manufacturing technique in integrated circuit fabrication, signal processing, and magnetic and optical storage. Although the underlying concepts have been firmly in place since the 1920s, commercialization of digital audio was postponed until the 1980s simply because theory had to wait 60 years for technology to catch up. Digital audio technology's complexity is all the more reason to start our discussion with some basics. Although this book deals mainly with digital topics, we must include at least one analog topic—sound. Once the nature of sound is understood, we can begin to explore ways to encode the information contained in an audio event and process and store it digitally.

Physics of Sound

It would be a mistake for a study of digital audio technology to forget the acoustic phenomena for which such a technology has been designed. Music is an acoustic event. Whether it originates from instruments radiating in air or from the direct creation of electrical signals, all music ultimately finds its way into the air, where it becomes a matter of sound and hearing. It is therefore appropriate to briefly review the fundamentals of the characteristics of sound, to establish a common understanding of its nature.

Sound Waves

Acoustics is the study of sound. As such, it is concerned with the generation, transmission, and reception of sound waves. The circumstances for these three phenomena are created when energy causes a disturbance in a medium. For example, when a kettledrum is struck, its drumhead disturbs the surrounding air (the medium). The outcome of that disturbance in air is the sound of a kettledrum. The mechanism is simple. The drumhead is activated and it vibrates back and forth. When the drumhead pushes forward, air molecules in front of it are condensed; when it pulls back, that area is rarefied. With the kettledrum, the drumhead's vibration is also affected by the air inside the drum's body, which acts as an air spring. In either case, the disturbance consists of regions of pressure above and below the equilibrium atmospheric pressure, as shown in figure 1-1. Nodes define areas of minimum displacement, while anti-nodes are areas of maximum (positive or negative) displacement.

The sound is propagated by air molecules through successive displacements that correspond to the original disturbance. In other words, air molecules colliding one against the next propagate the energy disturbance away from the source. Sound transmission thus consists of local disturbances propagating from one region to the next. A similar situation occurs when a rock is thrown into a pond; ripples spread across the surface of the pond, but the water itself stays fairly stationary. The local displacements of air molecules occur in the direction in which the disturbance is traveling; thus sound undergoes a longitudinal form of transmission. A receptor (like a microphone diaphragm) placed in the sound field will similarly move according to the pressure impinging on it, completing the chain of events. Incidentally, the more dense the medium, the easier is the task of propagation. For example, sound travels more easily in water than in air.

We can access an acoustic system with transducers, devices able to change energy from one form to another. These serve as sound generators and receivers. For example, a kettledrum changes the mechanical energy contributed by the mallet into acoustic energy. A microphone responds to the

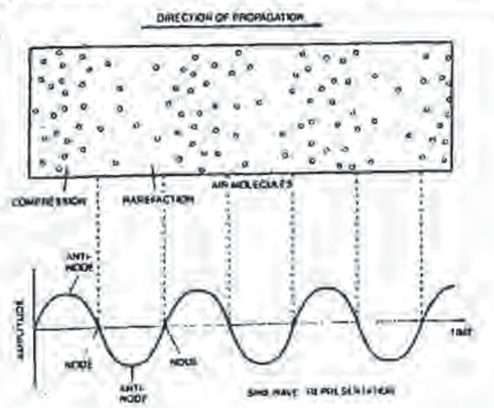


Fig. 1-1. Sound propagates through a medium with local displacement of molecules, as pressure maxima and minima.

acoustic energy by producing electrical energy. A loudspeaker reverses that process to again create acoustic energy from electrical energy.

The pressure changes of sound vibrations may be produced either periodically or aperiodically. A violin playing concert A moves the air back and forth periodically at a fixed rate. (In practice, things like vibrato make it a quasi-periodic vibration.) However, a cymbal crash has no fixed period; it is aperiodic. In the study of music, periodic vibrations are those most widely considered. One sequence of a periodic vibration, from pressure rarefaction to compression and back again, determines one cycle, or period. The number of vibration cycles that pass a given point each second is the frequency of the sound wave, measured in hertz (Hz). A violin playing concert A, for example, generates a waveform that repeats about 440 times per second; its frequency is 440 Hz. Alternatively, the reciprocal of frequency, the time it takes for one cycle to occur, is called the period. Frequencies can range from very low, such as changes in barometric pressure around 0.00001 Hz, to very high, such as cosmic rays at 10^{22} Hz. Sound is loosely described to be within the narrow, low frequency band from 0 Hz to 20 kHz—roughly the range of human hearing. Digital audio devices, as we shall see, are designed to respond to frequencies only in that range.

Wavelength is the distance sound travels through one complete cycle of pressure change, and is the physical measurement of the length of one cycle, as shown in figure 1-2. Because the velocity of sound is relatively constant (about 1,130 feet/second) we may calculate the wavelength of a sound wave by dividing the velocity of sound by its frequency. Quick calculations demonstrate the enormity of the differences in the wavelength of sounds. For example, a 20 kHz signal is about 0.7 inch long, while a 20 Hz signal is about 56 feet long. Few transducers (including our ears) are able to linearly receive or produce that range of wavelengths. Their frequency response is not flat, and the frequency range is limited. The range between the lowest and highest frequencies that a system can accommodate defines a system's bandwidth. To quantify that measurement, the deviation from flat response is specified according to application. Figure 1-3 illustrates an audio device with flat response from 60 Hz to 9 kHz. However, its bandwidth might be specified as 20 Hz to 20 kHz.

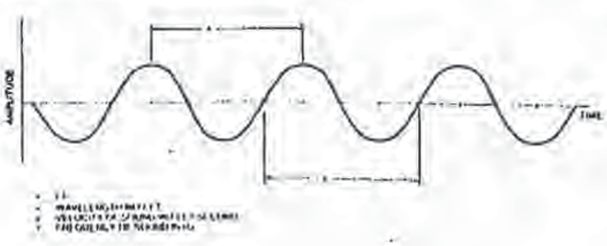


Fig. 1-2. A periodic waveform is characterized in terms of its frequency and wavelength, and the speed of sound.

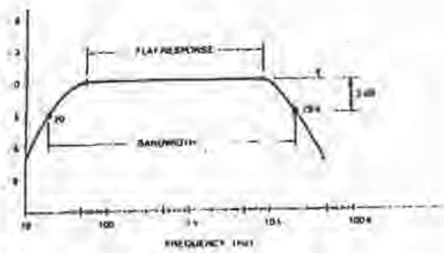


Fig. 1-3. Bandwidth measures a signal's useful frequency response.

Amplitude

Volume is a question of pressure amplitude—that is, the amount of pressure displacement above and below the equilibrium atmospheric level. Sound pressure is the instantaneous sound pressure minus the equilibrium (usually atmospheric) pressure. Sound pressure is very small because the amount of particle displacement in a medium is very small. In normal conversation, particle displacement is only about one-millionth of an inch, while a crowd's acoustic outpouring may cause displacement of about one-thousandth of an inch. In terms of actual numbers, if atmospheric pressure is 15 pounds/square inch, a loud sound might cause a deviation from 14.999 to 15.001 pounds/square inch. The range from the softest to the loudest sound determines the dynamic range. Because ears (and hence audio systems) have a dynamic range spanning a factor of millions, a logarithmic ratio is used to measure sound pressure levels.

The characteristics of sound require a measuring unit able to accommodate the large range of values we encounter in electrical and acoustic systems. The decibel (dB) uses base 10 logarithmic units to achieve this. A base 10 logarithm is the power to which 10 must be raised to equal the value. For example, an unwieldy number such as 100,000,000 yields a logarithm of 8 ($10^8 = 100{,}000{,}000$). Specifically, the dB is defined to be one-tenth the logarithm of a power ratio, as demonstrated in the following:

$$\text{Level} = 10 \log \frac{P_1}{P_2}\ \text{dB}$$

where:

P_1 and P_2 are values of acoustic or electrical power.

If the dominator of the ratio is set to a reference value, standard measurements may be made. For example, if $P_r = 0.001$ watt is the reference,

and a microphone's output is measured to be 0.0000001 watt, this is equivalent to:

$$\begin{aligned} \text{Power level} &= 10 \log \frac{P_1}{P_r}\ \text{dB} \\ &= 10 \log \frac{10^{-7}}{10^{-3}} \\ &= 10 \log 10^{-4} \\ &= (10)(-4) \\ &= -40\ \text{dB} \end{aligned}$$

In acoustic measurements, intensity levels (IL) may be measured in dB by setting the reference intensity to 10^{-12} watts/m² (threshold of hearing). Thus the intensity level of a rock band producing a sound of 10 watts/m² may be calculated:

$$\begin{aligned} \text{Intensity level} &= 10 \log \frac{P_1}{P_r}\ \text{dB} \\ &= 10 \log \frac{10^{1}}{10^{-12}} \\ &= 10 \log 10^{13} \\ &= (10)(13) \\ &= 130\ \text{dB SPL} \end{aligned}$$

When ratios of currents, voltages, or sound pressures are used (quantities whose square is proportional to power), the decibel formula becomes:

$$\text{Level} = 20 \log \frac{P_1}{P_r}\ \text{dB}$$

The zero reference level for acoustic sound pressure level measurement is a pressure of 0.0002 dyne/cm². This corresponds to the threshold of human hearing, the lowest SPL we can perceive, which is equal to 0 dB SPL. The threshold of feeling, the loudest level before discomfort begins, is rated at 120 dB SPL. All sound pressure levels that we normally perceive may be rated on a scale in terms of SPL, as shown in figure 1-4. A quiet home might have an SPL of 35 dB, a busy street might be 70 dB SPL, and the sound of a jet engine in close proximity might exceed 150 dB SPL. An orchestra's pianissimo might be 30 dB SPL, but a fortissimo might be 110 dB SPL; thus its dynamic range is 80 dB.

The logarithmic nature of these decibels should be considered. They are not commonly recognizable, insofar as they are not linear measurements. If two motorcycle engines, each producing an IL of 80 dB, were started together, the combined IL would not be 160 dB. Rather, the logarithmic result would be a 3 dB increase, yielding a combined IL total of 83 dB. Of course, in terms of linear units, those two motorcycles each producing sound

Fig. 1-4. Sound pressure levels—dB SPL.

intensities of 0.0001 watt/m² would combine to produce 0.0002 watt/m². You're right. It's confusing.

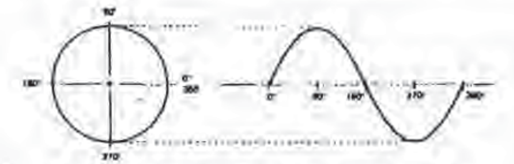
Phase

Two waveforms, identical in both shape and amplitude, may in fact be quite different because of their phase. If they are delayed relative to each other in time, phase shift occurs. Phase shift is often measured in degrees, as shown in figure 1-5. If two waveforms are combined and their relative phase altered, a new waveform results from constructive and destructive interference. Phase shift is thus the effect of relative time delay with respect to two signals, either acoustically or electrically analog in nature. It can alter the nature of waveforms; hence it can cause distortion. Phase distortion can result from the subsequent summation of phase-shifted signals, or can be induced in one signal as frequency-dependent phase shift. For example, if the high frequency content of a signal is delayed with respect to the low frequencies, the higher harmonics of the signal will arrive after the fundamental frequency. Absolute time delay holds no such danger. For example, the time delay between the making of a studio recording and playback in the home is quite long, but irrelevant to the waveform's structure, at least in terms of absolute time.

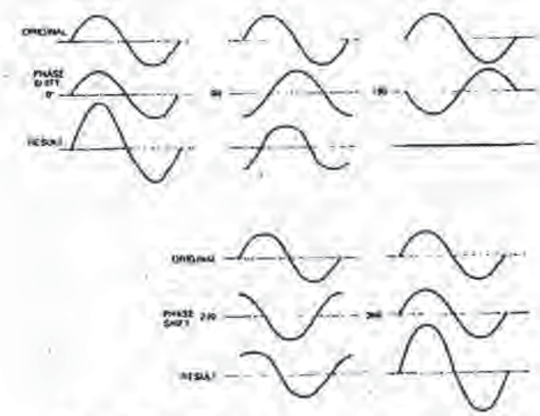
Complex Waveforms

The simplest form of periodic motion is the sinewave. It is manifested by the simplest oscillators, such as pendulums and tuning forks. The sinewave is unique because it exists only as a fundamental frequency. All other periodic waveforms are complex and are comprised of a fundamental frequency and a series of other frequencies at multiples of the fundamental, usually with decreasing amplitude. On the other hand, aperiodic complex waveforms, such as the sound of a motorcycle engine, do not exhibit this relationship. Most musical instruments are examples of the special case in which the harmonics are related to the fundamental through simple multiples. For example, a complex pitched waveform with a 150 Hz fundamental will have

Fig. 1-5. Phase shift can be characterized as displacement through a waveform's period.



(A) One period is divided into 360°, in both a circle and a periodic waveform.

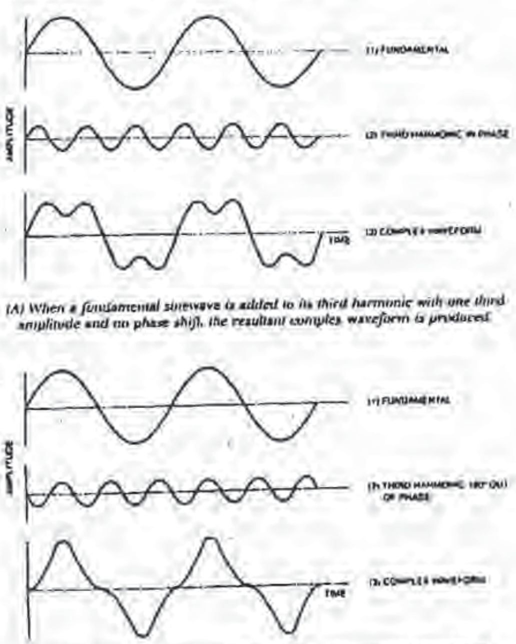


(B) Phase shift between two waveforms produces a new waveform through cancellation and reinforcement.

overtones at 300 Hz, 450 Hz, 600 Hz, 750 Hz, etc. When a string is plucked, it vibrates at its fundamental pitch over the length of the string. Simultaneously, it vibrates over its half-length, producing a quieter harmonic an octave higher, and over other lengths to produce overtones at the musical fifth, the second octave, third, seventh, etc.

Overtones extend through the upper reaches of human hearing. Twenty kHz is almost six octaves above concert A and is thus well beyond the range of any acoustic instrument's fundamental. However, it is the relative amplitudes of those overtones and their phase relationships that account for the waveform's timbre. For example, when the third harmonic is added to the fundamental, a complex waveform results. If the third harmonic is displaced in time (phase shifted) and added to the fundamental, yet another complex waveform is created, as demonstrated in figure 1-6. The two complex waveforms shown would have the same pitch, yet sound dissimilar. For example, a cello and trumpet may both play a note with the same fundamental pitch; however, their timbres are obviously different because of their differing harmonic series. This explains why the ear rapidly loses the ability to distinguish timbre of high frequency sounds. The first overtone of a 10 kHz tone is at 20 kHz; many people have trouble perceiving that overtone, let alone others that are even higher in frequency. Still, to properly record a complex waveform, both its fundamental and harmonic structure must be preserved, at least up to the limit of hearing. The harmonic nature of periodic waveforms is summarized by the Fourier Theorem. It states that all complex periodic waveforms are comprised of a harmonic series of sinewaves; thus, for example, we may synthesize complex waveforms by summing sinewaves. Furthermore, we may decompose a complex waveform into its sinewave content to analyze the nature of the complex waveform.

Fig. 1-6. Complex periodic waveforms are composed of harmonically related sinewaves.



(A) When a fundamental sinewave is added to its third harmonic with one third amplitude and no phase shift, the resultant complex waveform is produced.

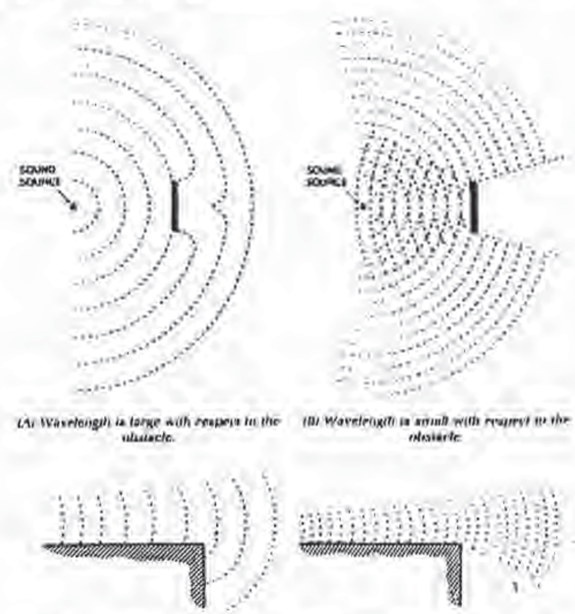
(B) When the third harmonic is shifted 180°, a new complex waveform is produced.

String players are well familiar with the practical applications of that theorem. When a cellist is asked to play the note D4 as a natural harmonic, he or she bows the open D string, which normally produces a note of pitch D3, while touching the string at its midpoint. The pitch is raised by an octave because the player has damped out all the odd-numbered harmonics, including the fundamental. The pitch changes; moreover, since the harmonic structure is changed, the timbre is changed as well.

Other Phenomena

Sound can undergo diffraction, in which it bends around obstacles, as shown in figure 1-7. Diffraction is relative to wavelength—longer wavelengths diffract more apparently than shorter ones. Thus, high frequencies are considered to be more directional in nature. Try this experiment: Hold a magazine in front of a loudspeaker—high frequencies will be blocked by the barrier, while longer wavelengths will go around it.

Fig. 1-7. Sound diffracts around obstacles, contingent on the relative wavelength.



(A) Wavelength is large with respect to the obstacle. *(B) Wavelength is small with respect to the obstacle.*

(C) Long wavelengths bend readily. *(D) Short wavelengths do not bend readily.*

2 Digital Basics

Whereas acoustics and analog audio technology are mainly concerned with continuous mathematical functions that represent the waveform, digital audio is a study of discrete values. Specifically, a waveform's amplitude is represented as a series of numbers. That is an important first principle, because numbers allow us to manage audio information very efficiently. Using digital computer techniques, our capability to process this information has been greatly enhanced. The design nature of audio recording, signal processing, and reproducing hardware has followed the advance of digital technology and, for the first time, the idea of programming has been introduced into the practical audio environment. Thus, digital audio is primarily a numerical technology. To properly understand it we must first establish some groundwork with a discussion of number systems, focusing on the binary system used in digital computers.

The Binary Number System

The basic problem confronting any digital audio system is the representation of audio information in digital form. While many possibilities present themselves, the only logical choice is the binary number system. This base 2 representation is ideally suited for storing and processing numerical information. Fundamental arithmetic operations are facilitated, as are logic operations.

The Meanings of Numbers

It all begins with numbers. When we deal with audio we are dealing with information, and numbers—as opposed to analog representation—offer a fabulous way to code, process, and decode information. In digital audio we use numbers to entirely represent audio information. We usually think of numbers as symbols, and the numerical symbols themselves are highly versatile; their meaning can vary according to the way we use them.

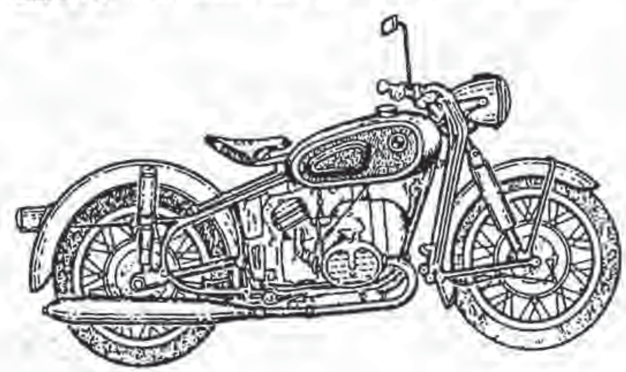
For example, consider my classic BMW R50/2 motorcycle, built in 1962 with a 500 cc engine and license plate 129907, shown in figure 2-1. Obviously, there are many numbers here; not so obvious is the important context of each. R50/2 represents the motorcycle's model number; 1962 is the year of manufacture; and the number 500 represents the quantity of cubic centimeters of engine displacement. The license number represents still another kind of information, specifically, coded information that allows my speeding tickets to be properly credited to my account. These various numbers are useful only by virtue of their arbitrarily assigned contexts. If that context is confused, then information encoded by the numbers goes awry. I could end up with a motorcycle with license number 1962, manufactured in the year 500, with an engine displacement of 129907 cubic centimeters.

Similarly, the numerical operations we perform upon numbers are matters of interpretation. The tally of my moving violations determines when my license will be suspended, but the sum of my license plate numerals is probably harmless. Numbers, if properly defined, provide a good method for storing and processing data. The negative implication is that numbers and their meanings have to be used carefully.

Number Systems

For most of us, the most familiar numbers are those of the base 10 system, perfected in the ninth century by some clever Arabs who conceived of the "0" numeral to represent nothing and appended it to the nine other numerals already in use. Earlier societies were stuck with the unitary system, which used one symbol in a series of marks to answer the essential question—how

Fig. 2-1.
A BMW R50/2 motorcycle



many? That is an unwieldy system for large numbers; thus, higher base systems were devised. The Mesopotamians, who considered themselves to be a fairly advanced bunch, invented a number system that used 60 symbols. It was a little cumbersome, but even today, 3,700 years later, we still use the essence of their system to divide an hour into 60 minutes, a minute into 60 seconds, and a circle into 360 degrees.

A number system is essentially a question of preference, because any integer may be expressed using any base. Choosing a number system involves the question of how many different symbols you think is most convenient. Our base 10 system uses 10 numerals; we say that the radix of the system is 10. In addition, the system uses positional notation; the position of the numerals tells us the quantities of ones, tens, hundreds, thousands, etc. In other words, the number in each next place is multiplied by the next higher power of the base. A base 10 system is convenient for 10-fingered organisms such as humans, but other number bases may be more appropriate for other applications. Of course, you have to know the radix; the numerals "10" in base 10 represent the total number of fingers we have, but "10" in base 8 is the number of fingers minus the thumbs. Similarly, would you rather have 10,000 dollars in base 6, or 100 in base 60? Table 2-1 compares four number systems.

Base 2

Gottfried Wilhelm von Leibnitz, the great philosopher and mathematician, stumbled onto the binary number system on March 15, 1679. That day marks the origin of today's digital systems. While base 10 is handy for humans, a base 2, or binary, system is more efficient for digital computers and digital audio equipment. Only two numerals are required to efficiently satisfy the

Table 2-1.
Four common number systems

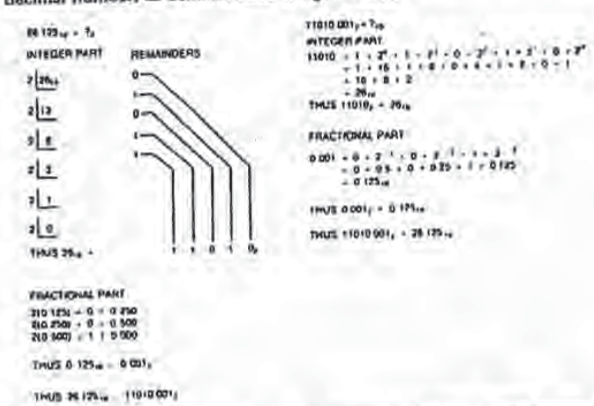
Hexadecimal (Base 16)	Decimal (Base 10)	Octal (Base 8)	Binary (Base 2)
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	0111
8	8	10	1000
9	9	11	1001
A	10	12	1010
B	11	13	1011
C	12	14	1100
D	13	15	1101
E	14	16	1110
F	15	17	1111

machine's principal electrical concern of *voltage/no voltage* or *on/off*. A binary system is ruthlessly efficient for a machine, and it is fast. Imagine how quickly you can turn a switch on and off; that represents the *rate* at which you can process information. Or watch a square wave go by. That means a machine is operating the switch for you. And consider the advantages in terms of storage. Instead of saving infinitely different analog values, you only have to remember two values. As we will see, the efficiency of binary data enables digital circuits to handle the tremendous amount of information contained in an audio signal.

Whatever information is being processed—in our case, whatever kind of audio signal has been converted to binary data—no matter how unrelated it might appear to be to numbers, a digital processor (or computer) codes the information in the form of a number, using the base 2 system. To better understand how audio data is handled inside the digital audio system, a brief look at the arithmetic of base 2 will be useful. In fact, we will consistently see that the challenge of coding audio information in digital form is a central issue in the design and operation of digital audio systems.

In essence, all number systems perform the same function; thus, we may familiarize ourselves with the binary system by comparing it to our decimal system. A given number may be expressed in terms of either base system and converted from one base to another. Several methods may be used. An easy decimal-to-binary conversion for whole numbers is division of the decimal number by 2 and collection of the remainders to form the binary number, as demonstrated in figure 2-2(A). Similarly, conversion from binary to decimal can be accomplished by writing the expression for the binary number in power of 2 notation, then expanding and collecting terms to form the decimal number, as demonstrated in figure 2-2(B).

Fig. 2-2. Base 10/base 2 conversion



(A) Conversion of a decimal number to binary

(B) Conversion of a binary number to decimal

This points up the fact that the base 2 system also employs positional notation. In base 2, each next place represents a doubling of value. The rightmost column represents 1s, the next column is 2s, then 4s, 8s, 16s, etc. Once again, it is important to designate the base being used; for example, in base 2 the symbol "10" could represent the total number of hands we have.

Just as a decimal point is used to delineate the whole number from the fractional number, a binary point does the same for binary numbers. Conversion of the fractional part of a decimal number to a binary number is done by multiplying the decimal number by 2. Conversion often leads to an infinitely sustaining binary number, so we must limit the number of terms.

As in our base 10 system, the standard arithmetic operations of addition, subtraction, multiplication, and division are applicable to the base 2 systems, as shown below.

Addition:

$$\begin{array}{r}0\\+0\\\hline 0\end{array}\qquad\begin{array}{r}0\\+1\\\hline 1\end{array}\qquad\begin{array}{r}1\\+0\\\hline 1\end{array}\qquad\begin{array}{r}1\\+1\\\hline 10\end{array}$$

Carry

Subtraction:

Borrow

$$\begin{array}{r}1\\-1\\\hline 0\end{array}\qquad\begin{array}{r}1\\-0\\\hline 1\end{array}\qquad\begin{array}{r}0\\-0\\\hline 0\end{array}\qquad\begin{array}{r}10\\-1\\\hline 01\end{array}$$

Multiplication:

$$\begin{array}{r}0\\\times 0\\\hline 0\end{array}\qquad\begin{array}{r}0\\\times 1\\\hline 0\end{array}\qquad\begin{array}{r}1\\\times 0\\\hline 0\end{array}\qquad\begin{array}{r}1\\\times 1\\\hline 1\end{array}$$

Division:

$$\begin{array}{r}0\\\div 1\\\hline 0\end{array}\qquad\begin{array}{r}1\\\div 1\\\hline 1\end{array}$$

Examples of binary arithmetic:

$$\begin{array}{r}11\\00110\\+10110\\\hline 11100\end{array}\qquad\begin{array}{r}11011\\-01101\\\hline 01110\end{array}$$

DISH-Blue Spike-246
Exhibit 1010, Page 0789

$$\begin{array}{r} 1001 \\ \times 101 \\ \hline 1001 \\ 0000 \\ 1001 \\ \hline 101101 \end{array} \qquad \begin{array}{r} 1011 \\ 101\overline{)110111} \\ \underline{101} \\ 0011 \\ \underline{0000} \\ 111 \\ \underline{101} \\ 0101 \\ \underline{0101} \\ 0 \end{array}$$

Just as in any base, the fundamental operation—addition—is easily carried out in base 2 because we have memorized addition rules to form an addition table. In base 2, we are merely careful to use the addition rules unique to that base. The procedure is the same as in the decimal system, except that it's easier because the addition table is simpler. There are only four possible combinations, compared to the more than 100 possible combinations resulting from the rules of decimal addition. The generation of the carry, as in the decimal system, is necessary when the result is larger than the largest digit in the system. The algorithms for subtraction, multiplication, and division in the binary system are identical to the corresponding algorithms in the decimal system.

Thus, a number is what we make it, and the various systems—differing only by base—operate in about the same way. A computer's use of the binary system is merely a question of expediency; it presents no real barrier to our understanding of digital techniques. It is simply the most logical approach. Ask yourself, would you rather deal with 10, 60, an infinite analog number, or 2 voltage levels?

Binary Codes

Although the abstractions of binary mathematics do indeed form the basis of digital audio systems, the implementation of these primitives requires higher level processing. Specifically, the next step up the evolutionary ladder is the coding of binary information. For example, individual binary bits or numbers can be ordered into words with specific connotations attached. In this way, both symbolic and numerical information is more easily dealt with by digital systems.

Encoding Numbers

Just as the digits in a motorcycle's license number carry a specially assigned meaning, groups of binary numbers can be encoded with special information. For example, a decimal number can be converted directly to its equivalent binary value. The binary number is encoded as the binary representation of the original number. Obviously, there is a restriction on the number of possible values that can be encoded. Specifically, an n-bit binary number can encode 2^n numbers. Three bits, for example, could encode eight states: 000, 001, 010, 011, 100, 101, 110, and 111. These could correspond to the decimal numbers 0, 1, 2, 3, 4, 5, 6, and 7.

Negative numbers present a problem because the sign must be encoded (with bits) as well. For example, a 1 in the left-most place could designate a negative number and a 0 could designate a positive number. This kind of representation is called a signed-magnitude representation. Our 3-bit word might then correspond to the decimal numbers shown in table 2-2. One problem is the presence of both +0 and −0. As we shall see, other methods can be used to better represent negative numbers.

Because we live in a decimal world, it is often useful to create binary words coded to decimal equivalents, preserving the same kind of decimal characteristics. Unfortunately, we observe that there is no binary grouping that directly represents the ten decimal digits. Three bits handle the first seven decimal numbers and 4 bits handle sixteen. For greater efficiency, a more sophisticated coding method is desirable. This is easily accomplished with groups of 4 bits each, with each group representing a decimal digit:

$$\underbrace{a_3a_2a_1a_0}_{\text{First decimal digit}} \qquad \underbrace{b_3b_2b_1b_0}_{\text{Second decimal digit}} \qquad \underbrace{n_3n_2n_1n_0}_{n\text{th decimal digit}}$$

Given this system, there are many ways in which the ten decimal digits can be encoded as 4-bit binary words. Specifically, there are approximately 2.9×10^{10} possibilities. Given these choices, it makes sense to find a method that provides as many benefits as possible. For example, a good code should facilitate arithmetic operations and error correction, and minimize storage space and logic circuitry. Similarly, whenever digital audio designers select a coding method, they examine the same criteria.

Table 2-2. Signed magnitude binary representations of decimal numbers

Binary	Decimal
000	+0
001	+1
010	+2
011	+3
100	−0
101	−1
110	−2
111	−3

Weighted Codes

In many cases, weighted codes offer a number of advantages over the many other possibilities of representing numbers. In a weighted code, each binary bit is assigned a decimal value, called a weight. Each number represented by the weighted binary code is calculated from the sum of the weighted the answer is negative, but in two's complement form. Taking the two's complement and assigning a negative sign results in the number −10110. When performing two's complement subtraction, the final carry provides the sign of the result. A final carry of 1 indicates a positive answer, and a carry of 0 indicates a negative answer in its two's complement, positive form.

If base complementing seems like a lot of trouble, it is redeemed by its advantages when handling positive and negative (bipolar) numbers, which might, for example, represent an audio waveform. The most significant bit (MSB) is the sign bit. When it is 0, the number is positive; when it is 1, the number is negative. In true binary form, the number 5 may be represented by 00000101 and −5 by 10000101. By representing negative numbers in two's complement form, −5 becomes 11111011 and the sign is handled automatically. All additions and subtractions result in positive numbers in true binary form and negative numbers in two's complement form, with the MSB automatically in the proper sign form. Two's complement representation is the norm in digital signal processing. If confusing for humans, it's comforting for machines.

Boolean Algebra

The binary number system presents tremendous opportunities for the design of electronic equipment, including, of course, digital audio equipment. Boolean algebra is the method used to combine and manipulate binary signals. It is named in honor of its inventor, George Boole, who published his proposal for the system in 1854 in a very curious work entitled *An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities*. Incidentally, historians inform us that Boole's formal education ended in the third grade.

Boolean Operators

Boolean logic is essential to digital circuits because it provides the basis for decision making, logical operations, and condition testing. Using Boolean algebra, all logical decisions are performed with the binary digits 0 and 1, a set of operators, and a number of laws and theorems. The on/off nature of the system is ideally suited for realization in digital systems. With the set of fundamental logical operators to manipulate bits (or Binary digiTS), we have the tools necessary to design the logic circuits that comprise useful digital systems. Everything from hard-wired logic circuits to supercomputers may be designed by taking advantage of this efficient system.

The Boolean operators are shown in figure 2-4. The operators OR, AND, and EXCLUSIVE OR (XOR) combine two binary digits to produce a single-digit result. The Boolean operator NOT complements a binary digit. NAND and NOR are derived from the other operators. The operators may be used singly or in combinational logic to perform any possible logical operation.

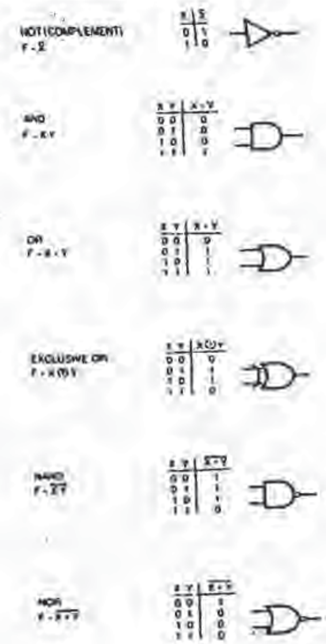


Fig. 2-4. Boolean operators may be used to manipulate logical conditions.

The complement, or NOT operation, complements any set of digits. The complement of 0 is 1 and the complement of 1 is 0. A bar is placed over the digit to represent a complement.

The AND operation is defined by the statement: If X AND Y are both 1, then the result is 1; otherwise the result is 0. Either a dot symbol or no symbol may be used to denote AND.

The OR operation is defined by the statement: If X OR Y, or both, are 1, then the result is 1; otherwise the result is 0. A plus sign is usually used to denote OR.

EXCLUSIVE OR differentiates binary states that are the same or different. Its output is 1 when X differs from Y, and is 0 when X is the same as Y. The XOR function thus represents binary addition. A circled plus sign is used to denote XOR.

Combining AND and NOT produces NAND, and combining OR and NOT produces NOR. Their results are the NOT of AND and OR, respectively.

Boolean Expressions

The Boolean operators can be combined into meaningful expressions giving statement to the condition at hand. Moreover, such statements often lead to greater insight into the condition, or to its simplification. For example, a digital system needs only the OR and NOT function, because any other function can be derived from those functions. This can be shown using De Morgan's Theorem, which states:

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$
$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

Using De Morgan's theorem, we observe that the expression:

$$A \cdot B = \overline{\overline{A} + \overline{B}}$$

generates AND from OR and NOT, and the expression:

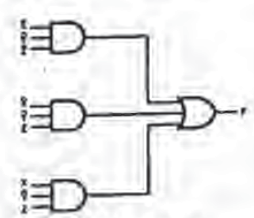
$$A \oplus B = \overline{(\overline{A} + B)} + \overline{(A + \overline{B})}$$

generates XOR from OR and NOT.

From this example, it should be clear that Boolean operators can be combined into expressions. In this case, De Morgan's Theorem is used to form the complement of expressions. It is the formulation of expressions that allows us to employ Boolean operators, along with one or more variables or constants, to solve applications problems. Parentheses are used to define the order of operations to be performed; operations are initiated within parentheses. When parentheses are omitted, complementation is performed first, followed by AND and then OR.

Moreover, logical expressions correspond directly to networks of logic gates, realizable in hardware. For example, figure 2-5(A) shows a logical expression and its equivalent network of logic gates. An expression could be evaluated by substituting a value of 0 or 1 for each variable, and carrying out the indicated operations. Each appearance of a variable or its complementation is called a literal. A truth table, or table of combinations, can be used to illustrate all the possible combinations contained in an expression.

Fig. 2-5. Boolean expressions may be realized in both hardware logic and truth tables.



$$F = \overline{X} \cdot \overline{Y} \cdot \overline{Z} + \overline{X} \cdot \overline{Y} \cdot Z + X \cdot \overline{Y} \cdot Z$$

(A) The hardware logic realization of a Boolean expression

X	Y	Z	$\overline{X} \cdot \overline{Y} \cdot \overline{Z}$	$\overline{X} \cdot \overline{Y} \cdot Z$	$X \cdot \overline{Y} \cdot Z$	F
0	0	0	1	0	0	1
0	0	1	0	1	0	1
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	1
1	1	0	0	0	0	0
1	1	1	0	0	0	0

(B) A truth table for a Boolean expression

In other words, the output can be expressed in terms of the input variables. For example, the truth table in figure 2-5(B) shows the results of the logic circuit in figure 2-5(A).

Boolean Theorems

Given a set of operators and ways to combine them into expressions, our next step is to develop a system of Boolean algebraic relations. That set forms the basis of digital processing in the same way that regular algebra governs the manipulation of our familiar base 10 operations. In fact, the base 2 and base 10 algebraic systems are very similar, to the point of confusion. Relations such as complementation, commutation, association, and distribution, as shown below, form the system of mathematical logic needed to create logical circuits and software. Remember that these laws hold true for Boolean algebra, but they are often uniquely defined.

Some of the principal laws of Boolean algebra

1. Special properties of 0 and 1
   $0 + X = X \qquad 0 \cdot X = 0$
   $1 + X = 1 \qquad 1 \cdot X = X$
2. Idempotence laws
   $X + X = X \qquad X \cdot X = X$
3. Involution
   $\overline{\overline{X}} = X$
4. Complementation laws
   $X + \overline{X} = 1 \qquad X \cdot \overline{X} = 0$
5. Commutative laws
   $X + Y = Y + X \qquad X \cdot Y = Y \cdot X$
6. Associative laws
   $X + (Y + Z) = (X + Y) + Z = X + Y + Z$
   $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z = X \cdot Y \cdot Z$
7. Distributive laws
   $X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$
   $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$
8. Absorption laws
   $X + (X \cdot Y) = X \qquad X \cdot (X + Y) = X$
   $X + (\overline{X} \cdot Y) = X + Y \qquad X \cdot (\overline{X} + Y) = X \cdot Y$

Double complementation results in the original value. In other words, when 1 is complemented it becomes 0, and when complemented again, it becomes 1 again. Commutative laws state that the order in which a combining of terms (with addition and multiplication operators) is performed does not affect its result. Associative laws state that when several terms are added or multiplied, the order of their selection for the operation is immaterial. Distributive laws demonstrate that the product of one term multiplied by a sum term equals the sum of the products of the first term multiplied by each product term.

3 Fundamentals of Digital Audio

The use of digital methods for the recording, reproduction, and storage of digital audio signals entails several concepts foreign to analog audio methods. In fact, digital audio systems bear little resemblance to analog systems, especially in terms of the critical function of each—the processing of audio information. Since audio itself is analog in nature, digital systems employ sampling and quantization, the twin pillars of audio digitization, to transform the audio information. Special precautions must be taken to combat two fundamental types of distortion: a condition of erroneous frequencies known as aliasing, and the error resulting from the quantization of the analog waveform.

Discrete Time Sampling

With analog recording we continuously modulate tape or cut a groove, but with digital we must use numbers. The first question we face is how to choose numbers. In other words, how do we record a data point from a changing waveform? Digitization employs time sampling and amplitude quantization to encode the infinitely variable analog waveform as discrete values in time and amplitude. We will consider these techniques in the following sections. First, let's consider the idea of sampling, the essence of digital audio.

The Lossless Nature of Sampling

Let's use a clock analogy to illustrate how sampling differentiates a digital music system from an analog system. Time seems to flow continuously. The hands of an analog clock sweep across the clock face covering all time as it passes by. A digital readout clock also tells time, but with a discretely valued display. In other words, it displays sampled time. It is the same with music. Music varies continuously in time and may be recorded and reproduced either in continuous analog form or time-sampled digital form. Just as both clocks tell the same time, both types of recording play the same music. Time sampling is the essential mechanism that defines a digital audio system, permits its analog-to-digital conversion, and differentiates it from an analog system.

However, a nagging question presents itself. If a digital system samples discretely, what happens between samples? Haven't we lost the information occurring between sample times? The answer, intuitively surprising, is no. Given correct conditions, no information is lost due to sampling between the input and output of a digitization system. The samples contain the same amount of information as the conditioned unsampled signal. To illustrate this, let's try another conceptual experiment.

Suppose we fasten a movie camera on the handlebars of our BMW motorcycle and go for a drive, up and down hills, over smooth pavement and some not so smooth, then return home and process the film. When we audition our piece of avant-garde cinema, we discover that the discrete frames of film successfully merge to reproduce our ride; it looks great. But when we come to some bumpy pavement, our picture is blurred. We ascertain that the quick movements were too fast for each frame to capture the change. We draw the following conclusions: If we increased the frame rate, using more frames per second, we would be able to capture quicker changes. If we complained to City Hall and had the bad pavement smoothed, then there would be no blur even at slower frame rates. Our movie would perfectly reproduce our motorcycle ride. We settle on a compromise—we make the roads reasonably smooth, then use a frame rate adjusted for a clean picture.

Just as the discrete frames of a movie create a moving picture, the samples of a digital audio recording create time-varying music. There is little conceptual difference between the visual and aural systems. Just as no useful information is lost between the frames of a properly shot motion picture, nothing is lost between the samples of a digital audio recording. As we discussed, sampling is a lossless process if the signal is properly conditioned. Thus, in a digital audio system, we must smooth out the bumps in the incoming signal. Specifically, the signal is low-pass filtered; that is, the frequencies too high to be properly sampled are removed. We design the system so that the threshold of these filtered frequencies is above the limit of human hearing.

The Sampling Theorem

When the input signal is low-pass filtered, we can theoretically sample the signal so that there is no loss of information (due to sampling) between the output sampled signal and the input smoothed signal. From a sampling standpoint, it is not an approximation; it is exact, as stated by Shannon and Nyquist. The discrete time method of sampling defines only instantaneous values. However, it can be mathematically proven that a sampled band-limited signal contains the same amount of information as the original unsampled band-limited signal. When the signal is smoothed, we can reconstruct all the

intervening values without error and thus re-create the original waveform. Consider the waveform in figure 3-1. The continually changing analog function has been sampled to create a series of pulses. The amplitude of each pulse, when chosen from a vertical scale, ultimately yields a number that represents the analog amplitude at that instant. To quantify the situation, we define the sampling frequency as the number of samples per second. Its reciprocal, sampling rate, is the time between each sample. For example, a sampling frequency of 40,000 samples/second corresponds to a rate of $1/40,000$ second. It is apparent that a quickly changing waveform—that is, one with high frequencies—would require a faster sampling frequency, as we saw in our motorcycle movie. Thus, sampling frequency determines the frequency

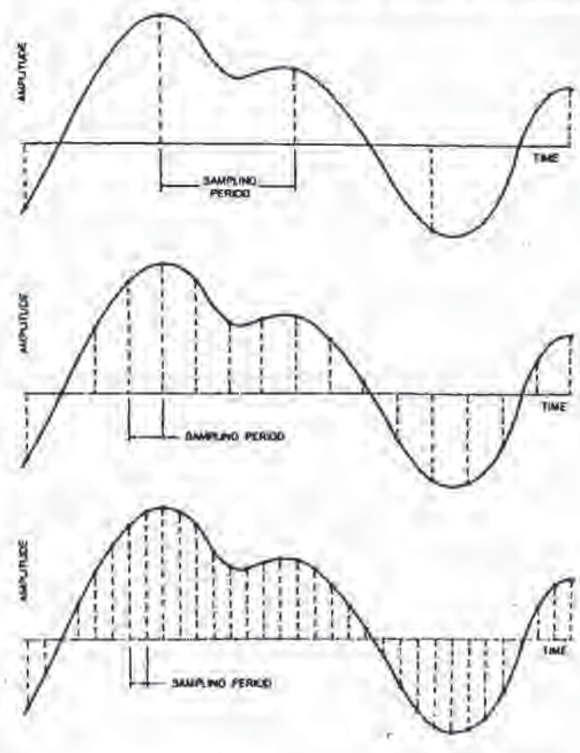


Fig. 3-1. Discrete time sampling divides a signal into equal, periodically spaced intervals. The higher the sampling rate, the smaller the sampling period.

response and overall signal throughput bandwidth of the digitization system. The choice of sampling frequency is one of the most important design criteria of a digitization system, because it determines bandwidth of the system. Thus, the question is presented—how often should we sample to accurately represent a music waveform?

Sampling theory described by Shannon and Nyquist answers the question of sampling frequency: S samples/second are needed to completely represent a waveform with a bandwidth of S/2 Hz. In other words, we must sample at a frequency of twice the highest audio frequency to achieve lossless sampling. For example, an audio signal with a frequency response of 0 to 20 kHz would theoretically require a sampling frequency of 40 kHz for proper digital encoding. It is crucial to observe the sampling theorem's criteria for limiting the input signal to no more than half the sampling frequency (a point sometimes called the Nyquist Frequency). Just as a bumpy road blurred our movie, too high an audio frequency in a digitization system would cause distortion. This is examined in greater detail later in this chapter. A low-pass filter always precedes the sampling circuit to remove frequencies above the half-sampling frequency limit. A low-pass filter is also placed at the output of a digital audio system to remove high frequencies created internal to the system. This filter smooths the staircase effect in the reconstructed sampled waveform to recover the original waveform, as shown in figure 3-2. This is discussed in more detail in Chapter 5.

Another question presents itself with respect to the sampling theorem. It can be observed that low audio frequencies can be easily sampled, because of their long wavelengths, there are many samples to represent each period. But as the sampled frequencies become higher, the periods are shorter and there are fewer samples per period. Finally, in the theoretical limiting case of critical sampling, at an audio frequency of half the sampling frequency, there are only two samples per period. However, even two samples can represent a waveform. For example, consider the case of a 40 kHz sampling system and an audio sinewave input at 20 kHz, as shown in figure 3-3. The digitizer would produce two samples, which would be used to construct a 20 kHz square wave. In itself, this reconstructed waveform is quite unlike the original sinewave. However, the 20 kHz square wave is comprised of odd harmonics—sinewaves at 20, 60, 100, 140, and 180 kHz, etc. A low-pass filter at the output of the digital audio system removes all frequencies higher than those originally entering the system. With all higher harmonics removed, the output of the system is a 20 kHz sinewave, the same waveform as originally input. We know that the 20 kHz input waveform was a sinewave because the input low-pass filter would not have passed higher waveform harmonics to the sampler. As far as our ears are concerned, a sinewave is perfectly suitable because the harmonics of any complex 20 kHz waveform are above our audible range anyway. The first partial, for example, would be located at 40 kHz. Thus, even in the limiting case, the sampling theory is valid. It is correct to state that higher sampling rates would permit recording and reproduction of higher audio frequencies. But given the design criteria of an audio frequency bandwidth, higher rates would not improve the fidelity of those frequencies already within the allowed frequency range.

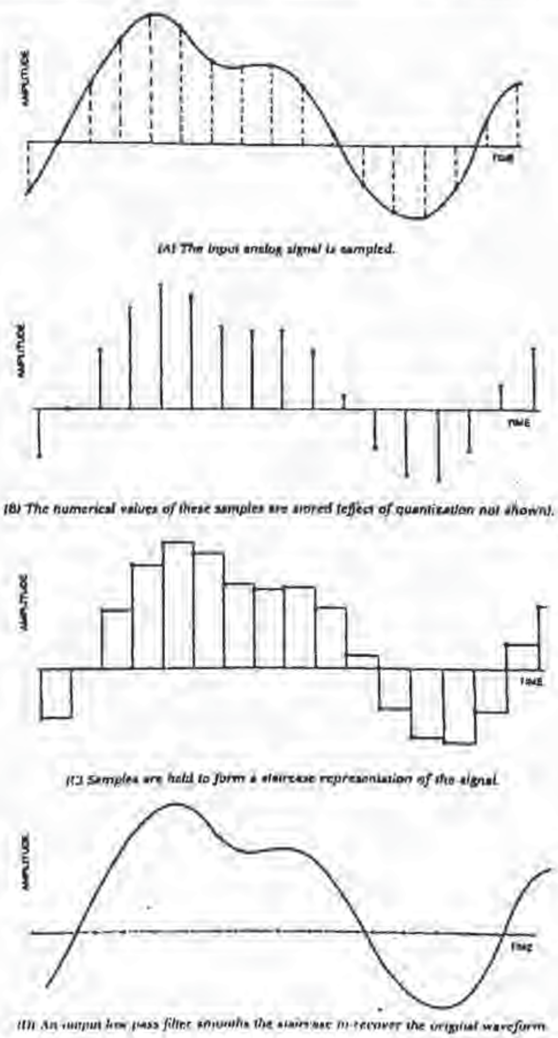


Fig. 3-2. With discrete time sampling, a band-limited signal can be sampled and reconstructed without loss due to sampling.

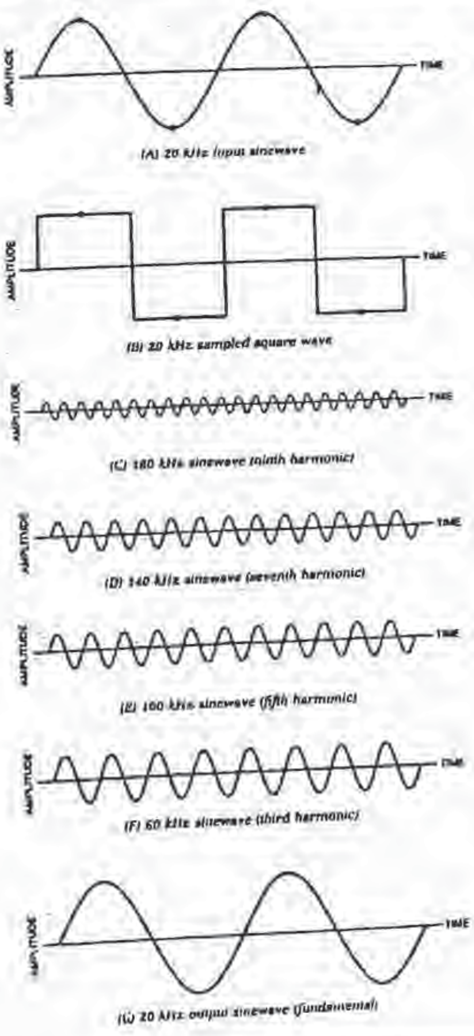
(A) The input analog signal is sampled.

(B) The numerical values of these samples are stored (effect of quantization not shown).

(C) Samples are held to form a staircase representation of the signal.

(D) An output low-pass filter smooths the staircase to recover the original waveform.

Fig. 3-3. When a 20 kHz sinewave is sampled at a 40 kHz rate, the two sample points reconstruct a square wave.



(A) 20 kHz input sinewave

(B) 20 kHz sampled square wave

(C) 180 kHz sinewave (ninth harmonic)

(D) 140 kHz sinewave (seventh harmonic)

(E) 100 kHz sinewave (fifth harmonic)

(F) 60 kHz sinewave (third harmonic)

(G) 20 kHz output sinewave (fundamental)

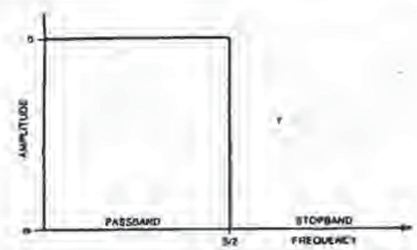
In the previous example of critical sampling, there is no guarantee that the sample times will coincide with the maxima and minima of the waveform. Samples could come from lower amplitude parts of the waveform, or even coincide with the zero axis crossings of the waveform. In practice, this poses no problem. Critical sampling is not attempted; a sampling margin is always present. As we have seen, to satisfy the sampling theorem, manufacturers must design a low-pass filter into any digitization system, placing it first in the signal chain. Because these analog filters cannot cut off the signal precisely at the frequency where the sampling theorem demands, a guard band is employed. The filter's cutoff frequency characteristic is begun at a lower frequency, allowing several thousand Hz for the filter to sufficiently attenuate the signal. This assures that no frequency higher than half the sampling rate enters the digitization circuitry.

It should be emphasized that the need to low-pass filter the audio signal is not as detrimental as it might first appear. As long as we choose an appropriate sampling rate, we can extend the frequency response of the audio signal as far as we wish. The trade-off, of course, is the demand we place on the speed of digital circuitry and the capacity of the storage medium. Higher sampling rates require that circuitry operate faster and that larger amounts of data be stored. Both are ultimately questions of economics. Manufacturers have chosen a sampling rate of 44.1 kHz for the compact disc, for example, because such a system can be affordably produced.

The entire sampling (and de-sampling) process is illustrated in figure 3-4. The signals involved in sampling are shown at various points in the chain. Moreover, the left half of the figure shows the signals in the time domain and the right half portrays the frequency domain. In other words, we can observe a signal's amplitude over time, as well as its frequency response. We observe in figure 3-4(A) and (B) that the input audio signal must be band-limited to the half-sampling frequency S/2, using an anti-aliasing filter. The sampling signal in figure 3-4(C) and (D) recurs at the sampling frequency S, and its spectrum consists of impulses at multiples of the sampling frequency. When the audio signal is sampled, as shown in figure 3-4(E) and (F), the signal's amplitude at sample time is preserved; however, this signal contains images of the original spectrum centered at multiples of the sampling frequency. To reproduce the sampled signal, as in figure 3-4(G) and (H), the samples are passed through an anti-imaging filter at S/2. This smoothes the de-sampled waveform, re-creating a smooth audio signal.

At any rate, the point is clear: a band-limited signal may be sampled, stored as discrete values, de-sampled, and reproduced. No information is lost through sampling. Sampling theorems, such as the Nyquist Theorem, demonstrate this conclusively. Of course, time sampling is only half the battle. A digital system must also determine the actual numerical values it will use at sample time to represent the original waveform's amplitude. This question of quantization is explained later in this chapter. Readers who consider this explanation of sampling to be too elementary are invited to see the appendix.

Fig. 3-11.
An ideal low-pass filter characteristic attenuates infinitely at S/2.



It is critical to observe sampling theory and low-pass filter the input signal in a digitization system. If aliasing is allowed to occur, there is no technique that can remove the aliased frequencies from the original audio bandwidth. As we shall see, extremely low level aliasing can occur after the anti-aliasing filter, because of quantization error. A noise signal called dither is used to alleviate this distortion. A mathematical analysis of frequency spectra, as shown in figure 3-12, summarizes the effects of aliasing. In figure 3-12(A) the sampling theorem is observed, whereas in figure 3-12(B) it is not observed, with aliasing as a result.

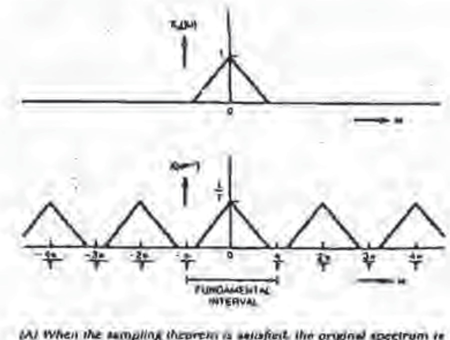
Quantization

To record an audio signal, two dimensions of information must be stored. Sampling implicitly saves time information and quantization saves amplitude information. Quantization is thus the measured value of the analog signal at sample time. With quantization, as with the measurement of any analog event, accuracy is limited by the system's resolution. Because of finite word length, a digital system's resolution is limited, and a measuring error is introduced. This error is similar to noise in an analog system; however, it differs because its character changes with signal amplitude.

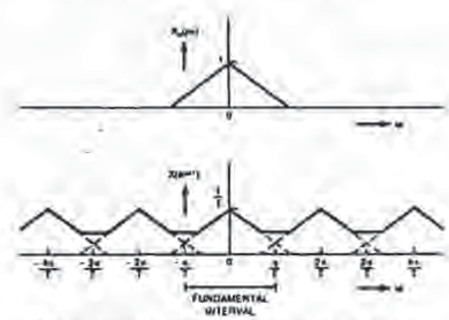
Analog and Digital Approximation

Let's use an example to illustrate the effects of quantization and differentiate its error from the error inherent in an analog system. Suppose we have connected two voltmeters, one analog and one digital, as shown in figure 3-13. At the final chord of the Beethoven Ninth, we read both meters, measuring the voltage corresponding to the acoustic input signal. Given a good meter face and a sharp eye, we read the analog needle at 1.27 volts. A rather cheap digital meter may have only two digits and thus we would read 1.3 volts. If we had paid a little more for a three-digit meter we might have read 1.27 volts and a four-digit meter might have read 1.274 volts. Now, both the analog and digital meters are always in error. The error in the analog meter is due

Fig. 3-12.
The spectral effects of aliasing (Courtesy Philips Technical Review)



(A) When the sampling theorem is satisfied, the original spectrum is repeated.



(B) When aliasing occurs, repetitions overlap and the original spectrum can never be completely recovered.

to the ballistics of the mechanism and our difficulty in reading the meter. Even under ideal conditions, at some point any analog measurement capacity is lost in the device's own noise.

With the digital meter, the nature of the error is different. Accuracy is limited by the resolution of the meter—that is, by the number of digits displayed. The more digits, the greater the accuracy, but the last digit will always round off relative to the actual value; for example, 1.27 was rounded off to 1.3 in the cheap meter. Under the best conditions, the last digit would be completely accurate; for example, a voltage of exactly 1.3000 would be shown

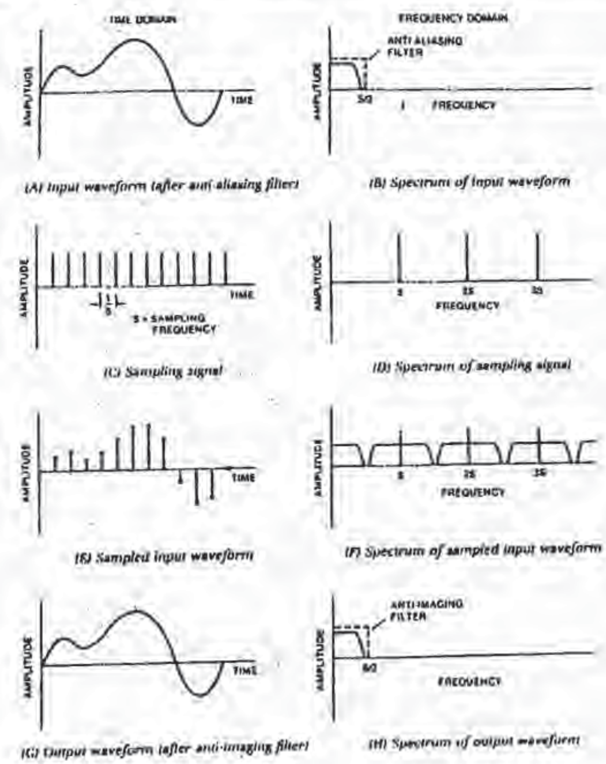
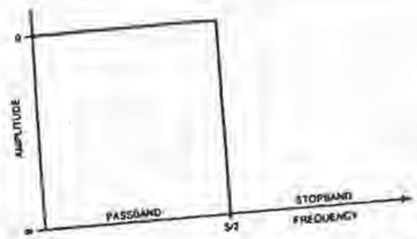


Fig. 3-4. Time domain (left side) and frequency domain (right side) signals illustrate the effect of band-limited waveform sampling and reconstruction.

Fast Sampling Rates

Before we close the book on discrete time sampling, we should mention a current hypothesis concerning the nature of time. We mentioned that time seems to be continuous. However, some physicists have recently suggested that, like energy and matter, time might come in discrete packets. Just as this book consists of a finite number of atoms and could be converted into a finite amount of energy, the time it takes you to read the book might consist of a finite number of time particles. Specifically, the indivisible period of time might be 1×10^{-42} second (that's a 1 preceded by a decimal point and 41 zeros). The theory is that no time interval can be shorter than this, because

Fig. 3-11. An ideal low-pass filter characteristic attenuates infinitely at S/2.



It is critical to observe sampling theory and low-pass filter the input signal in a digitization system. If aliasing is allowed to occur, there is no technique that can remove the aliased frequencies from the original audio bandwidth. As we shall see, extremely low level aliasing can occur after the anti-aliasing filter, because of quantization error. A noise signal called dither is used to alleviate this distortion. A mathematical analysis of frequency spectra, as shown in figure 3-12, summarizes the effects of aliasing. In figure 3-12(A) the sampling theorem is observed, whereas in figure 3-12(B) it is not observed, with aliasing as a result.

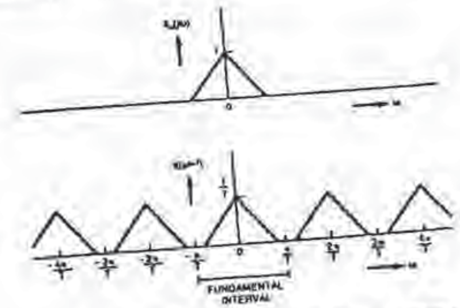
Quantization

To record an audio signal, two dimensions of information must be stored. Sampling implicitly saves time information and quantization saves amplitude information. Quantization is thus the measured value of the analog signal at sample time. With quantization, as with the measurement of any analog event, accuracy is limited by the system's resolution. Because of finite word length, a digital system's resolution is limited, and a measuring error is introduced. This error is similar to noise in an analog system; however, it differs because its character changes with signal amplitude.

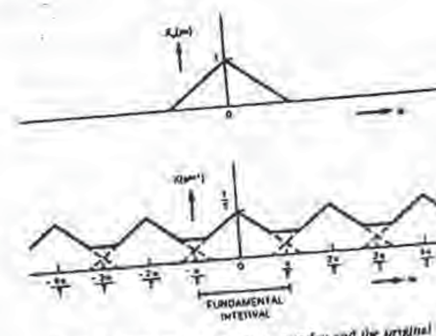
Analog and Digital Approximation

Let's use an example to illustrate the effects of quantization and differentiate its error from the error inherent in an analog system. Suppose we have connected two voltmeters, one analog and one digital, as shown in figure 3- ... of the Beethoven Ninth, we read both meters, measuring ... input signal. Given a good meter ...

Fig. 3-12. The spectral effects of aliasing (Courtesy Philips Technical Review)



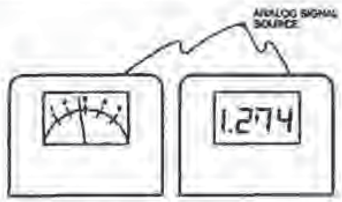
(A) When the sampling theorem is satisfied, the original spectrum is repeated.



(B) When aliasing occurs, repetitions overlap and the original spectrum can never be completely recovered.

to the ballistics of the mechanism and our difficulty in reading the meter. Even under ideal conditions, at some point any analog measurement capacity is lost in the device's own noise.

With the digital meter, the nature of the error is different. Accuracy is limited by the resolution of the meter—that is, by the number of digits displayed. The more digits, the greater the accuracy, but the last digit will always round off relative to the actual value; for example, 1.27 was rounded off to ... Under the best conditions, the last digit would be ... 1.3000 would be shown



(A) Analog voltmeter (B) Digital voltmeter

Fig. 3-13. Approximation in measurement implies an error in attempting to express the actual value.

as 1.3. Under the worst condition, the rounding off will be one-half interval away; for example, 1.250 would be rounded off to 1.2 or 1.3. If a binary system is used for the measurement, we say that the error resolution of the system is one-half the least significant bit (LSB). For both analog and digital systems, the problem of measuring an analog phenomenon such as amplitude leads to error. At least as far as voltmeters are concerned, a digital read-out is an inherently more robust kind of measurement; we gain more information about an analog event when it is characterized in terms of digital data. The analog voltmeter has gone the way of the slide rule.

Quantization is thus the technique of measuring an analog event to form a numerical value. Of course, a digital system usually dictates the use of a binary number system. In terms of the quantizing hardware, the number of possible values is determined by the length of the data word—that is, the number of bits available to form the representation. Just as the number of digits in our digital voltmeter determined our resolution, the number of bits in our digitization equipment determines resolution. As we shall see, that decision is primarily influenced by the quality of the analog-to-digital (A/D) converter.

Approximation in Measurement

The task of recording and reproducing music can be simply summarized: we want to form a representation of the music. The closer our representation is to the original, the better. Unfortunately, reality is stubborn in its ability to defy re-creation, and we are left with a challenging endeavor as we attempt to create an approximation of the original event by saving as much information as possible.

The essential problem lies in the complexity of even the simplest acoustic waveform and the dual nature of the information it carries. No matter which recording system we employ, to characterize an acoustic event we must correlate time and amplitude information. Thus, a vinyl LP has a groove, the length of which implicitly encodes time, and lateral variations, which encode amplitude. In a digital system, both time (implicitly again) and amplitude are stored as discrete pieces of information.

We have discussed sampling, a method of periodically taking a measurement. Of course, taking a measurement of a varying event is meaningful

only if both the time and the value of the measurement are stored. Sampling represents the time of the measurement, and quantization represents the value of the measurement, or in the case of audio, the amplitude of the waveform at sample time. Sampling and quantization are thus the fundamental components of digitization, and together, at least in theory, can characterize any acoustic event. Both sampling and quantization become variables that determine, respectively, the bandwidth and resolution of the representation. An originally analog waveform may be mapped into a series of pulses; the amplitude of each pulse yields a number that represents the analog value at that instant.

The interplay between sampling rate and quantization is shown in figure 3-14. Correct sampling of a band-limited signal is a lossless process, but choosing the amplitude value at sample time is not. Any choice of scales or codes, as shown in table 3-1, results in the realization that digitization can never totally encode a continuous analog function. An analog waveform has an infinite number of amplitude values, whereas we can only choose from a finite number of intervals. All of the analog values between two intervals can only be represented by the single number assigned to that interval. Thus, our chosen value is only an approximation of the actual. In other words, with quantization, there is an error.

Signal-to-Error Ratio

With a binary number system, the word length determines the number of quantizing intervals available; this can be computed by raising the word length to the power of 2, as shown in the box below.

Number (N) of quantization intervals in a binary word $N = 2^n$ where n is the number of bits in the word.

$2^1 = 2$	$2^7 = 128$	$2^{13} = 8192$	$2^{19} = 524288$
$2^2 = 4$	$2^8 = 256$	$2^{14} = 16384$	$2^{20} = 1048576$
$2^3 = 8$	$2^9 = 512$	$2^{15} = 32768$	$2^{21} = 2097152$
$2^4 = 16$	$2^{10} = 1024$	$2^{16} = 65536$	$2^{22} = 4194304$
$2^5 = 32$	$2^{11} = 2048$	$2^{17} = 131072$	$2^{23} = 8388608$
$2^6 = 64$	$2^{12} = 4096$	$2^{18} = 262144$	$2^{24} = 16777216$

Thus, an 8-bit word would accommodate $2^8 = 256$ intervals and a 16-bit word would provide $2^{16} = 65{,}536$ intervals. The more bits, the better the approximation; but as we have seen, there must always be an error associated with quantization, because the limited number of amplitude choices contained in the binary word can never completely accommodate an infinite number of analog possibilities. At some point, the quantizing error becomes audibly indistinguishable. Most manufacturers have agreed that 16 to 20 bits

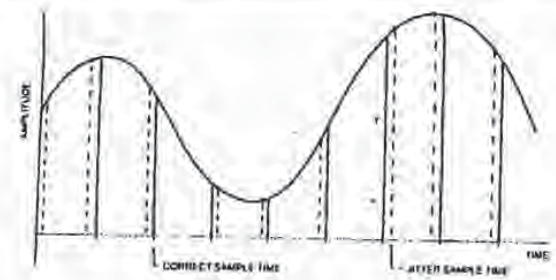


Fig. 4-17. Time axis variations in sampling cause jitter.

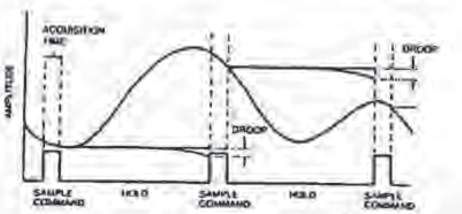


Fig. 4-18. Two error conditions in the sample and hold circuit: acquisition time and droop.

can be corrected; however, the delay is a function of the amplitude of the analog signal. The best solution is prevention; therefore, it is important to limit the acquisition timing error. Jitter is discussed further in Chapter 7.

The S/H circuit's other primary function is to hold the captured analog voltage while conversion takes place. It is important for this voltage to remain constant, because any variation greater than a quantization increment will result in an error on the A/D output. In practice, the held voltage is prone to droop because of current leakage. Droop is the variation in hold voltage as the capacitor slowly leaks between samples. Care in circuit design and selection of components can limit droop to less than one-half a quantization increment over a 20-microsecond period. For example, a 16-bit, ±10-volt range A/D converter would require holding a constant value to within 1 millivolt during conversion. Acquisition error and droop are illustrated in figure 4-18. Acquisition time is the time between the initiation of the sample command and the taking of the sample.

Sample and Hold Circuit Design

The demands of fast acquisition time and low droop are sometimes in conflict in the design of an S/H circuit. For fast acquisition time, a small capacitor value is better, permitting faster charging time in response to the hold com-

mand. For droop, however, a large valued capacitor is preferred, because it is better able to retain the sample voltage at a constant level for a longer time. Circuit designers have found that capacitor values of approximately 1 nanofarad can satisfy both requirements. In addition, high quality capacitors made of polycarbonate, polyethylene, or Teflon dielectrics are specified. These materials can respond quickly, hold charge, and minimize dielectric absorption and hysteresis—phenomena that cause voltage variations.

In practice, an S/H circuit must contain more than a switch and a capacitor. Active circuits such as operational amplifiers must buffer the circuit to condition the input and output signals, speed switching time, and help prevent leakage. Only a few specialized operational amplifiers meet the required specifications of large bandwidth and fast settling time. Junction Field Effect Transistor (JFET) operational amplifiers usually perform best. Thus, a complete S/H circuit might have a JFET input operational amplifier to prevent source loading and speed switching time, isolate the capacitor, and supply capacitor charging current. The S/H switch itself is probably a JFET device, selected to operate cleanly and accurately with minimal jitter, and the capacitor is high quality. A JFET operational amplifier is usually placed at the output to help preserve the capacitor's charge. An example of a practical sample and hold circuit is shown in figure 4-19. Switch A is closed to sample. After conversion, switch B is closed to discharge capacitor C and prepare for another sample.

The sample and hold circuit thus time samples and stores analog values for conversion. Its output signal is an intermediate signal, a discrete staircase of the original analog signal, but still not a digital word.

Analog-to-Digital Conversion

The analog-to-digital converter lies at the heart of the recording side of a PCM audio digitization system, and it is the most critical and costly component in the entire electronic system. This circuit must determine which quantization interval is closest to the analog waveform's current value, and output a binary number specifying that level, accomplishing that task in 20 microseconds or less. Fortunately, several types of circuits are available for this operation. Two fundamental analog-to-digital design approaches prevail. The input analog voltage can be compared to a variable reference voltage within a feedback loop to determine the output digital word, or the input

Fig. 4-19. An example of a practical sample and hold circuit with JFET switches