PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04L

A2

(11) International Publication Number: WO 96/42151

(43) International Publication Date: 27 December 1996 (27.12.96)

(21) International Application Number: PCT/US96/10257

(22) International Filing Date: 7 June 1996 (07.06.96)

(30) Priority Data:
08/489,172 9 June 1995 (09.06.95) US

(71) Applicant: THE DICE COMPANY [US/US]; P.O. Box 60471, Palo Alto, CA 94306-0471 (US).

(72) Inventors: COOPERMAN, Marc, S.; 2929 Ramona, Palo Alto, CA 94306 (US). MOSKOWITZ, Scott, A.; Townhouse 4, 20191 East Country Club Drive, North Miami Beach, FL 33180 (US).

(74) Agents: ALTMILLER, John, C. et al.; Kenyon & Kenyon, 1025 Connecticut Avenue, N.W., Washington, DC 20036 (US).

(81) Designated States: CA, CN, FI, JP, KR, SG, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published

Without international search report and to be republished upon receipt of that report.

(54) Title: STEGANOGRAPHIC METHOD AND DEVICE

(57) Abstract

An apparatus and method for encoding and decoding additional information into a stream of digitized samples in an integral manner. The information is encoded using special keys. The information is contained in the samples, not prepended or appended to the sample stream. The method makes it extremely difficult to find the information in the samples if the proper keys are not possessed by the decoder. The method does not cause a significant degradation to the sample stream. The method is used to establish ownership of copyrighted digital multimedia content and provide a disincentive to piracy of such material.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	GB	United Kingdom	MW	Malawi
AT	Austria	GE	Georgia	MX	Mexico
AU	Australia	GN	Guinea	NE	Niger
BB	Barbados	GR	Greece	NL	Netherlands
BE	Belgium	HU	Hungary	NO	Norway
BF	Burkina Faso	IE	Ireland	NZ	New Zealand
BG	Bulgaria	IT	Italy	PL	Poland
BJ	Benin	JP	Japan	PT	Portugal
BR	Brazil	KE	Kenya	RO	Romania
BY	Belarus	KG	Kyrgystan	RU	Russian Federation
CA	Canada	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic			SE	Sweden
CG	Congo	KR	Republic of Korea	SG	Singapore
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LR	Liberia	SZ	Swaziland
CS	Czechoslovakia	LT	Lithuania	TD	Chad
CZ	Czech Republic	LU	Luxembourg	TG	Togo
DE	Germany	LV	Latvia	TJ	Tajikistan
DK	Denmark	MC	Monaco	TT	Trinidad and Tobago
EE	Estonia	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	UG	Uganda
FI	Finland	ML	Mali	US	United States of America
FR	France	MN	Mongolia	UZ	Uzbekistan
GA	Gabon	MR	Mauritania	VN	Viet Nam

STEGANOGRAPHIC METHOD AND DEVICE

Definitions

Several terms of art appear frequently in the following. For ease of reference they are defined here as follows:

"Content" refers to multimedia content. This term encompasses the various types of information to be processed in a multimedia entertainment system. Content specifically refers to digitzed audio, video or still images in the context of this discussion. This information may be contained within files on a multimedia computer system, the files having a particular format specific to the modality of the content (sound, images, moving pictures) or the type of systems, computer or otherwise, used to process the content.

"Digitized" refers to content composed of discrete digital samples of an otherwise analog media, which approximate that media inside a computer or other digital device. For instance, the sound of music occurs naturally, and is experienced by humans as an analog (continuous) sound wave. The sound can be digitized into a stream of discrete samples, or numbers, each of which represents an approximate

value of the amplitude of the real analog wave at a particular instant in time. These samples can be stored in files in a computer and then used to recreate the original sound wave to a high degree of accuracy.
In general, content entering a digital system is digitized by Analog to Digital converters (A/D) and analog media are recreated by the digital system using a Digital to Analog (D/A) converter. In the context of this discussion content is always digitized content.

"Cryptography" is a field covering numerous techniques for scrambling information conveying messages so that when the message is conveyed between the sender and receiver an unintended party who intercepts this message cannot read it, or extract useful information from it.

A "Public Key Cryptosystem" is a particular cryptographic system where all parties possess pairs of keys for encryption and decryption. Parties to this type of system freely distribute their public keys, which other may use to encrypt messages to the owner of the public key. Such messages are decrypted by the receiver with the private key. Private keys are never distributed. A message encrypted with a public key can only be decrypted with the corresponding private key, and vice versa. A message encrypted with a private key is said to have been signed by the owner of that key. Anyone in possession of the public key may decrypt the message and know that it was encrypted, and thus signed, by the owner of the public key, since only they possess the corresponding private key.

"Steganography" is a field distinguished from cryptography, but associated with it, that covers numerous methods for hiding an informational message within some other medium, perhaps another unrelated message, in such a manner that an unintended party who intercepts the medium carrying the hidden message does not know it contains this hidden message and therefore does not obtain the information in the hidden message. In other words, steganography seeks to hide messages in plain view.

Background of the Invention

In the current environment of computer networks and the proliferation of digital or digitized multimedia content which may be distributed over such networks, a key issue is copyright protection. Copyright protection is the ability to prevent or deter the proliferation of unauthorized copies of copyrighted works. It provides a reasonable guarantee that the author of a copyrighted work will be paid for each copy of that work.

A fundamental problem in the digital world, as opposed to the world of physical media, is that a unlimited number of perfect copies may be made from any piece of digital or digitized content. A perfect copy means that if the original is comprised of a given stream of numbers, then the copy matches the original, exactly, for each number in the stream. Thus, there is no degradation of the original signal during the copy operation. In an analog copy, random noise is always introduced, degrading the copied signal.

The act of making unlicensed copies of some content, digital or analog, whether audio, video, software or other, is generally known as *piracy*. Piracy has been committed for the purpose of either profit from the sale of such unlicensed copies, or to procure for the "pirate" a copy of the content for personal use without having paid for it.

The problem of piracy has been made much worse for any type of content by the digitization of content. Once content enters the digital domain, an unlimited number of copies may be made without any degradation, if a pirate finds a way to break whatever protection scheme was established to guard against such abuses, if any. In the analog world, there is generally a degradation in the content (signal) with each successive copy, imposing a sort of natural limit on volume of piracy.

To date, three general types of schemes have been implemented in an attempt to protect copyrights.

1) Encryption
2) Copy Protection
3) Content Extensions

Copy Protection and Content Extensions generally apply in the digital world only, while a scheme related to Encryption, commonly known as scrambling, my be applied to an analog signal. This is typical in analog cable systems.

Encryption scrambles the content. Before the content is made ready for delivery, whether on floppy disk, or over a network, it must be encrypted, or scrambled. Once the content has been encrypted, it cannot be used until it is decrypted, or unscrambled. Encrypted audio data might sound like incomprehensible screeching, while an encrypted picture or video might appear as random patterns on a screen. The principle of encryption is that you are free to make as many copies as you want, but you can't read anything that makes sense until you use a special key to decrypt, and you can only obtain the key by paying for the content.

Encryption has two problems, however. 1) Pirates have historically found ways to crack encryption, in effect, obtaining the key without having paid for it; and 2) Once a single legitimate copy of some content has been decrypted, a pirate is now free to make unlimited copies of the decrypted copy. In effect, in order to sell an unlimited quantity of an encrypted piece of software, the pirate could simply buy one copy, which they are entitled to decrypt.

Copy Protection includes various methods by which a software engineer can write the software in a clever manner to determine if it has been copied, and if so to deactivate itself. Also included are undocumented changes to the storage format of the content. Copy protection was generally abandoned by the software industry, since pirates were generally just as clever as the software engineers and figured out ways to modify their software and deactivate the protection. The cost of developing such protection was not justified considering the level of piracy which occurred despite the copy protection.

Content Extension refers to any system which attaches some extra information to the original content which indicates whether or not a copy may be made. A software or hardware system must be specifically built around this scheme to recognize the additional information and interpret it in an appropriate manner. An example of such a system is the Serial Copyright Management System embedded in Digital Audio Tape (DAT) hardware. Under this system, additional information is stored on the disc immediately preceding each track of audio content which indicates whether or not it can be copied. The hardware reads this information and uses it accordingly.

A fundamental problem with Encryption and Content Extension is the "rogue engineer". An employee who helped design such a system or an individual with the knowledge and means to analyze such a system can modify it to ignore the copyright information altogether, and make unlicensed copies of the content. Cable piracy is quite common, aided by illicit decoder devices built by those who understand the technical details of the cable encryption system. Although the cable systems in question were actually based on analog RF signals, the same principle applies to digital systems.

The practical considerations of weak encryption schemes and rogue engineers have served to limit the faith which may be put in such copyright protection schemes. The invention disclosed herein serves to address these problems with conventional systems for digital distribution. It provides a way to enforce copyright online. The invention draws on techniques from two fields, cryptography, the art of scrambling messages so that only the intended recipient may read them, and steganography, a term applied to various techniques for obscuring messages so that only the intended parties to a message even know that a message has been sent, thus it is termed herein as a stega-cipher. The stega-cipher is so named because it uses the steganographic technique of hiding a message in multimedia content, in combination with multiple keys, a concept originating in cryptography. However, instead of using the keys to encrypt the content, the stega-cipher uses these keys to locate the hidden message within the content. The message itself is encrypted which serves to further protect the message, verify the validity of the message, and redistribute the information in a random manner so that anyone attempting to locate the message without the keys cannot rely on pre-supposed knowledge of the message contents as a help in locating it.

Summary of the Invention

The invention disclosed herein combines two techniques, steganography - obscuring information that is otherwise in plain sight, and cryptography - scrambling information that must be sent over unsecured means, in a manner such that only the intended recipient may successfully unscramble it. The net effect of this system is to specifically watermark a piece of content so that if it is copied, it is possible to determine who owned the original from which the copies were made, and hence determine responsibility for the copies. It is also a feature of the system to uniquely identify the content to which it is applied.

For a comprehensive discussion of cryptography, its theory, applications and specific algorithms, see APPLIED CRYPTOGRAPHY, by Bruce Schneier, which is herein incorporated by reference at pages 66-68, 387-392.

Steganography is discussed briefly in THE CODE BREAKERS by David Kahn, which is herein incorporated by reference at pages xiii, 81-83, 522-526, and 873. An example application, Stego by Romana Machado, is also available for the Apple Macintosh. Stego can be found at the internet uniform resource locator "ftp://sumex-aim.stanford.edu/info-mac/cmp/stego10a2.hqx". This application demonstrates a simple steganographic technique to encode a text message into a graphical image without significantly distorting the image.

The invention improves upon the prior art by providing a manner for protecting copyright in the digital domain, which neither steganography or cryptography does. It improves specifically on steganography by making use of special keys which dictate exactly where within a larger chunk of content a message is to be hidden, and makes the task of extracting such a message without the proper key the equivalent of looking for a needle in a haystack.

The information encoded by the Stega-Cipher process serves as a watermark which identifies individual copies of content legally licensed to specific parties. It is integral with the content. It cannot be removed by omission in a transmission. It does not add any overhead to signal transmission or storage. It does allow the content to be stored to and used with traditional offline analog and digital media, without modification or significant signal degradation. These aspects of the stega-cipher all represent improvements to the art. That is, its forces would - be pirates to damage the content in order to guarantee the disabling of the watermark.

The invention described herein is used for protecting and enforcing copyrights in the digital or on-line domain, where there are no physical limitations on copying copyrighted content.

The invention uniquely identifies every copy of multimedia content made using the invention, composed of digitized samples whether compressed or uncompressed, including but not limited to still digital images, digital audio, and digital video.

The invention is for use in meterware or pay-by-use systems where an online user incurs a charge each time they access a particular piece of content, or uses a software title.

The invention is for use as a general improvement to cryptographic techniques to increase the complexity of cryptanalysis on a given cipher.

It is considered that the method and steps of the present invention will be modified to account for the effects of loss compression schemes on the samples and particularly includes modification to handle MPEG compressed audio and video.

It is considered that statistical data spreading and recovery techniques, error coding or spread spectrum processing techniques might be applied in the invention to handle the effects of loss compression, or counter the effects of a randomization attack.

It is considered that the apparatus described might be further specialized and optimized in hardware by replacing general purpose data buses and CPU or DSP driven operations with hardwired circuitry, incorporated in one or more special purpose ICs.

It is considered that the apparatus will be modeled and implemented in software on general purpose computer platforms.

It is considered that stega-cipher hardware could be embedded in a consumer electronics device and used to not only identify content and copyright, but to enable use of that content.

Detailed Description

I. Digital Copyright Stega-Cipher Protocol and the Decode/Encode Program

The purpose of the program described here is to watermark digital multimedia content for distribution to consumers through online services in such a way as to meet the following criteria

Given a unique piece of multimedia content, composed of digitized samples, it is desirable to:

1) Uniquely identify this particular piece of content from others in a manner which is secure and undeniable (e.g. to know whether a digital audio recording is "My Way" by Frank Sinatra, or "Stairway to Heaven", by Led Zeppelin), and in a manner such that this identification can be performed automatically by an electronic device or mechanism.

2) Uniquely identify the copyright owner of the content, and the terms under which it may be distributed in general, in a manner which is secure and undeniable.

3) At such time as is necessary, additionally, uniquely identify in a secure and undeniable manner the licensed publisher who received a particular copy of the content, and the terms under which they may redistribute or resell it.

4) At such time as is necessary, additionally, uniquely identify in a secure and undeniable manner, the licensed subscriber who received a particular copy of the content from the publisher described in item 3.

The program described in more detail below combines the techniques of cryptography and steganography to hide a securely encrypted digital copyright certificate which contains information satisfying the criteria listed above, in such a manner as to be integral with the content, like a watermark on paper, so that possession of the content dictates possession of the watermark information. In addition, the watermark cannot be "found" or successfully decoded, without possession of the correct "masks" or keys, available only to those legitimately authorized, namely, those parties to a commercial transaction involving the sale of a copy of the content. Finally, the ability to distribute such watermarked content in a system which implements the watermark scheme is denied without a successfully decoded watermark. Because well known and tested cryptographic techniques are used to protect the certificate itself, these certificates are virtually impossible to forge. Finally, the watermark cannot be erased without significantly damaging the content.

The basic program represents a key part of the invention itself. This program is then used as the method by which copyright information is to be associated in an integral manner with the content. This is a concept absent from copy protection, encryption and content extension schemes. The copyright information itself can be made undeniable and unforgeable using cryptographic techniques, so.that through it an audit trail of ownership my be established for each copy of a given piece of content, thus customizing each copy to a particular owner, in a way that can be used to identify the owner.

The value of the stega-cipher is that it provides a way to watermark the content in a way that changes it slightly, but does not impact human perception significantly. And, furthermore, that it is made difficult to defeat since one must know exactly where the information resides to extract it for analysis and use in forgery attempts, or to remove it without overly degrading the signal. And, to try to forge copyright information one must first be able to analyze the encrypted copyright information, and in order to do that, one must be able to find it, which requires masks.

II. Example Embodiment of General Processing

Digital audio data is represented by a series of samples in 1 dimension,

$$\{S_1, S_2, S_3 \ldots S_n\}$$

This series is also referred to as a sample stream. The sample stream approximates an analog waveform of sound amplitude over time. Each sample represents an estimate of the wave amplitude at the instant of time the sample is recorded. For monaural audio, there is one such sample stream. Stereo audio is comprised of two sample streams, one representing the right channel, and the other representing the left. Each stream is used to drive a corresponding speaker to reproduce the stereo sound.

What is referred to as CD quality audio is characterized by 16 bit (2 byte) stereo samples, recorded at 44.1 Khz, or 44,100 samples per second in each channel. The dynamic range of sound reproduction is directly proportional to the number of bits per sample. Some lower quality recordings are done at 8 bits. A CD audio recording can be stored using any scheme for containing the 2 sample streams in their entirety. When these streams are played back at the same frequency they were recorded at, the sound recorded is reproduced to a high degree of accuracy.

The sample stream is processed in order from first sample to last. For the purpose of the invention disclosed, the stream is separated into sample windows, each of which has a fixed number of consecutive samples from the stream, and where windows do not overlap in the sample stream. Windows may be contiguous in the sample stream. In this discussion assume each window contains 128 samples, and that windows are contiguous. So, the windows within the stream look like

$$\{[S_1, S_2, S_3 \ldots S_{128}], [S_{129}, S_{130}, S_{131} \ldots S_{256}], \ldots [S_{n-128} \ldots S_n]\}$$

where [...] denotes each window and any odd samples at the end of the stream which do not completely fill a window can be ignored, and simply passed through the system unmodified.

These windows will be used as input for the discrete Fast Fourier Transform (and its inverse) operation.

Briefly, Fourier Transform methods are based on the principle that a complex waveform, expressed as amplitude over time and represented by a sample stream, is really the sum of a number of simple waveforms, each of which oscillate at different frequencies.

By complex, it is meant that the value of the next sample is not easily predicted from the values of the last N samples or the time of the sample. By simple it is meant that the value of the sample is easily predictable from the values of the last N samples and/or the time of the sample.

The sum of multiple simple waves is equivalent to the complex wave. The discrete FFT and its inverse simply translate a limited amount of data from one side of this equivalence to the other, between the complex waveform and the sum of simple waves. The discrete FFT can be used to translate a series of samples representing amplitude over time (the complex wave, representing a digital audio recording) into the same number of samples representing total spectral energy in a given range of frequencies (the simple wave components) at a particular instant of time. This instant is the time in the middle of the original amplitude/time samples. The inverse discrete FFT translates the data in the other direction, producing the complex waveform, from its simpler parts.

Each 128 sample window will be used as an input to the discrete FFT, resulting in 128 bins representing each of 128 frequency bands, ranging from 0Hz to 22Khz (the Nyquist frequency, or ½ the sampling rate).

Information can be encoded into the audio signal in the frequency domain or in the time domain. In the latter case, no FFT or inverse FFT is necessary. However, encoding in the frequency domain is recommended, since its effects are scattered over the resultant time domain samples, and not easily predicted. In addition, frequency domain encoding makes it more likely that randomization will result in noticeable artifacts in the resultant signal, and therefore makes the stega-cipher more defensible against such attacks. It is in the frequency domain that additional information will be encoded into the audio signal for the purpose of this discussion. Each frequency band in a given time slice can potentially be used to store a small portion of some additional information to be added to the signal. Since these are discrete estimates, there is some room for error which will not significantly effect the perceived quality of the signal, reproduced after modification, by the inverse FFT operation. In effect, intentional changes, which cannot be distinguished from random variations are introduced in the frequency domain, for the purpose of storing additional information in the sample stream. These changes are minimized so as not to adversely affect the perceived quality of the reproduced audio signal, after it has been encoded with additional information in the manner described below. In addition, the location of each of these changes is made virtually impossible to predict, an innovation which distinguishes this scheme from simple steganographic techniques.

Note that this process differs from the Nagata, et al. patents, 4,979,210 and 5,073,925, which encode information by modulating an audio signal in amplitude/time domain. It also differs in that the modulations introduced in the Nagata process (which are at very low amplitude and frequency relative to the carrier wave as to remain inaudible) carry only copy/ don't copy information, which is easily found and circumvented by one skilled in the art. Also, there is no limitation in the stega-cipher process as to what type of information can be encoded into the signal, and there is more information storage capacity, since the encoding process is not bound by any particular frequency of modulation but rather by the number of samples available. The granularity of encoding in the stega-cipher is determined by the sample window size, with potentially 1 bit of space per sample or 128 bits per window (a secure implementation will halve this to 64 bits). In Nagata, et al. the granularity of encoding is fixed by the amplitude and frequency modulation limits required to maintain inaudibility. These limits are relatively low, and therefore make it impractical to encode more than simple copy/ don't copy information using the Nagata process.

III. Example Embodiment of Encoding and Decoding

A modification to standard steganographic technique is applied in the frequency domain described above, in order to encode additional information into the audio signal.

In a scheme adapted from cryptographic techniques, 2 keys are used in the actual encode and decode process. For the purposes of this invention the keys are referred to as masks. One mask, the primary, is applied to the frequency axis of FFT results, the other mask is applied to the time axis (this will be called the convolution mask). The number of bits comprising the primary mask are equal to the sample window size in samples (or the number of frequency bands computed by the FFT process), 128 in this discussion. The number of bits in the convolution mask are entirely arbitrary. This implementation will assume a time mask of 1024 bits. Generally the larger the key, the more difficult it is to guess.

Prior to encoding, the primary and convolution masks described above are generated by a cryptographically secure random generation process. It is possible to use a block cipher like DES in combination with a sufficiently pseudo-random seed value to emulate a cryptographically secure random bit generator. These keys will be saved along with information matching them to the sample stream in question in a database for use in decoding, should that step become necessary.

Prior to encoding, some additional information to be encoded into the signal is prepared and made available to the encoder, in a bit addressable manner (so that it may be read one bit at a time). If the size of the sample stream is known and the efficiency characteristics of the stega-cipher implementation are taken into account, a known limit may be imposed on the amount of this additional information.

The encoder captures one sample window at a time from the sample stream, in sequential, contiguous order. The encoder tracks the sequential number of each window it acquires. The first window is 0. When the number of windows processed reaches the number of bits in the window mask, minus one, the next value of the window counter will be reset to 0.

This counter is the convolution index or phase. In the current implementation it is used as a simple index into the convolution bitmask. In anticipated developments it will be used to perform convolution operations on the convolution mask to determine which bit to use. For instance the mask might by rotated by a number corresponding to the phase, in bits to the left and XORed with the primary mask to produce a new mask, which is then indexed by the phase. There are many possibilities for convolution.

The encoder computes the discrete FFT of the sample window.

Starting with the lowest frequency band, the encoder proceeds through each band to the highest, visiting each of the 128 frequency bands in order. At each band value, the encoder takes the bit of the primary mask corresponding to the frequency band in question, the bit of the convolution mask corresponding to the window in question, and passes these values into a boolean function. This function is designed so that it has a near perfectly random output distribution. It will return true for approximately 50% of its input permutations, and false for the other 50%. The value returned for a given set of inputs is fixed, however, so that it will always return the same value given the same set of inputs.

If the function returns true, the current frequency band in the current window is used in the encoding process, and represents a valid piece of the additional information encoded in the signal. If the function returns false, this cell, as the frequency band in a given window is called, is ignored in the process. In this manner it is made extremely difficult to extract the encoded information from the signal without the use of the exact masks used in the encoding process. This is one place in which the stega-cipher process departs from traditional steganographic implementations, which offer a trivial decode opportunity if one knows the information is present. While this increases the information storage capacity of the carrier signal, it makes decoding trivial, and further degrades the signal. Note that it is possible and desirable to modify the boolean cell flag function so that it returns true < 50% of the time. In general, the fewer cells actually used in the encode, the more difficult they will be to find and the less degradation of content will be caused, provided the function is designed correctly. There is an obvious tradeoff in storage capacity for this increased security and quality.

The encoder proceeds in this manner until a complete copy of the additional information has been encoded in the carrier signal. It will be desirable to have the encoder encode multiple copies of the additional information continuously over the duration of the carrier signal, so that a complete instance of this information may be recovered from a smaller segment of a larger signal which has been split into discontinuous pieces or otherwise edited. It is therefore desirable to minimize the size of the information to be encoded using both compact design and pre-encoding compression, thus maximizing redundant encoding, and recoverability from smaller segments. In a practical implementation of this system it is likely the information will be first compressed by a known method, and then encrypted using public-key techniques, before being encoded into the carrier signal.

The encoder will also prepare the package of additional information so that it contains an easily recognizable start of message delimeter, which can be unique to each encoding and stored along with the keys, to serve as a synchronization signal to a decoder. The detection of this delimeter in a decoding window signifies that the decoder can be reasonably sure it is aligned to the sample stream correctly and can proceed in a methodic window by window manner. These delimeters will require a number of bits which minimizes the probability that this bit sequence is not reproduced in a random occurrence, causing an accidental misalignment of the decoder. A minimum of 256 bits is recommended. In the current implementation 1024 bits representing a start of message delimeter are used. If each sample is random, then each bit has a 50% probably of matching the delimeter and the conditional probability of a random match would be $1/2^{1024}$. In practice, the samples are probably somewhat less than random, increasing the probability of a match somewhat.

The decode process uses the same masks in the same manner, only in this case the information is extracted one bit at a time from the carrier signal.

The decoder is assumed to have access to the proper masks used to encode the information originally. These masks might be present in a database, which can be indexed by a value, or values computed from the original content, in a manner insensitive to the modifications to the content caused by the stega-cipher process. So, given an arbitrary piece of content, a decoder might first process the content to generate certain key values, and then retrieve the decode masks associated with the matching key values from the database. In the case where multiple matches occur, or none are found, it is conceivable that all mask sets in the database could be tried sequentially until a valid decode is achieved, or not, indicating no information is present.

In the application of this process, it is anticipated that encoding operations may be done on a given piece of content up to 3 times, each adding new information and using new masks, over a sub-segment of the content, and that decode operations will be done infrequently. It is anticipated that should it become necessary to do a search of a large number of masks to find a valid decode, that this process can be optimized using a guessing technique based on close key matching, and that it is not a time critical application, so it will be feasible to test large numbers of potential masks for validity on a given piece of content, even if such a process takes days or weeks on powerful computers to do a comprehensive search of known mask sets.

The decode process is slightly different in the following respect. Whereas the encoding process can start at any arbitrary point in the sample stream, the decode process does not know where the encode process began (the exact offset in samples to the start of the first window). Even though the encode process, by convention, starts with sample 0, there is no guarantee that the sample stream has not been edited since encoding, leaving a partial window at the start of the sample stream, and thus requiring the decoder to find the first complete window to start the decode. Therefore, the decode process will start at the first sample, and shift the sample window along by 1 sample, keeping the window index at 0, until it can find a valid decode delimeter encoded in the window. At this point, the decoder knows it has synchronized to the encoder, and can then proceed to process contiguous windows in a more expedient manner.

Example Calculations based on the described implementation for adding copyright certificate information to CD quality digital audio:

In a stream of samples, every 128 samples will contain, on average 64 bits of certificate related information. Digital audio is composed of 16 bit samples, at 44.1 Khz, or 44,100 samples per second. Stereo audio provides 2 streams of information at this rate, left and right, or 88,200 samples per second. That yields approximately 689 contiguous sample windows (of 128 samples) per second in which to encode information. Assume a song is 4 minutes long, or 240 seconds. This yields 240 * 689 = 165,360 windows, which on average (50% utilization) contain 64 bits (8 bytes) each of certificate information. This in turns gives approximately 1291Kb of information storage space per 4 minute stereo song (1.2 MB). There is ample room for redundant encoding of information continuously over the length of the content. Encoding 8 bytes for every 256 bytes represents 3.1% of the signal information. Assuming that a copyright certificate requires at most approximately 2048 bytes (2K), we can encode the same certificate in 645 distinct locations within the recording, or approximately every 37/100ths of a second.

Now to account for delimeters and synchronization information. Assuming a sync marker of 1024 bits to avoid random matches, then we could prefix each 2K certificate block with this 1024 bit marker. It takes 256 windows to store 2K, and under this proposed scheme, the first 16 windows are reserved for the sync marker. A decoder could search for this marker by progressively matching each of the first 16 windows (64 bits at a time) against the corresponding portion of the sync marker. The decoder could reset the match advancing through the sample stream, as soon as one window did not conform to the sync marker, and proceed in this manner until it matches 16 consecutive windows to the marker, at which point it is synchronized.

Under this scheme, 240 windows, or 1.92K remain for storing certificate information, which is not unreasonable.

IV. Possible Problems, Attacks and Subsequent Defenses

A. Randomization

The attacker simply randomizes the least significant bits of each data point in the transform buffer, obliterating the synchronization signal and the watermark. While this attack can remove the watermark, in the context in which stega-cipher is to be used, the problem of piracy is kept to a minimum at least equal to that afforded by traditional media, since the system will not allow an unwatermarked piece of content to be traded for profit and watermarks cannot be forged without the proper keys, which are computationally difficult to obtain by brute-force or cryptanalysis. In addition, if the encoding is managed in such a way as to maximize the level of changes to the sample stream to be just at the threshold below human perception, and the scheme is implemented to anticipate randomization attempts, it is possible to force the randomization level to exceed the level that can be perceived and create destructive artifacts in the signal, in much the same manner as a VHS cassette can be manufactured at a minimal signal level, so that a single copy results in unwatchable static.

B. Low Bit-Depth Bitmaps (black & white images)

These bitmaps would be too sensitive to the steganization process, resulting in unacceptable signal degradation, and so are not good candidates for the stega-cipher process. The problem may be circumvented by inflating bit-depth, although this is an inefficient use of space and bandwidth.

C. Non-Integer Transforms

The FFT is used to generate spectral energy information for a given audio signal. This information is not usually in integer format. Computers use methods of approximation in these cases to represent the real numbers (whole numbers plus fractional amounts). Depending on the exact value of the number to be represented slight errors, produced by rounding off the nearest real number that can be completely specified by the computer occur. This will produce some randomization in the least significant bit or bits. In other words, the same operation on the same sample window might yield slightly different transform values each time. It is possible to circumvent this problem using a modification to the simple LSB steganographic technique described later. Instead of looking at the LSB, the stega-cipher can use an energy quantization technique in place of the LSB method. Some variant of rounding the spectral energy values up or down, with a granularity greater than the rounding error should work, without significantly degrading the output samples.

V. A Method and Protocol For Using the Stega-Cipher

The apparatus described in the claims below operates on a window by window basis over the sample stream. It has no knowledge of the nature of the specific message to be encoded. It merely indexes into a bit stream, and encodes as many of those bits as possible into a given sample window, using a map determined by the given masks.

The value of encoding information into a single window in the sample stream using such an apparatus may not be inherently apparent until one examines the manner in which such information will be used. The protocol discussed in this section details how messages which exceed the encoding capacity of a single sample window (128 samples) may be assembled from smaller pieces encoded in the individual windows and used to defend copyrights in an online situation.

An average of 64 bits can be encoded into each window, which equals only 8 bytes. Messages larger than 8 bytes can be encoded by simply dividing the messages up and encoding small portions into a string of consecutive windows in the sample stream. Since the keys determine exactly how many bits will be encoded per window, and an element of randomness is desirable, as opposed to perfect predictability, one cannot be certain exactly how many bits are encoded into each window.

The start of each message is marked by a special start of message delimeter, which, as discussed above is 1024 bits, or 128 bytes. Therefore, if precisely 8 bytes are encoded per window, the first 16 windows of any useable message in the system described here are reserved for the start of message delimeter. For the encoder, this scheme presents little challenge. It simply designates the first sample window in the stream to be window 0, and proceeds to encode the message delimeter, bit-by-bit into each consecutive window. As soon as it has processed the last bit of the SOM delimeter it continues by encoding 32 bits representing the size, in bytes of the complete message to follow. Once the 32nd and final bit of the size is encoded, the message itself is encoded into each consecutive window, one bit at a time. Some windows may contain more encoded bits then others, as dictated by the masks. As the encoder processes each window in the content it increments its window counter. It uses this counter to index into the window mask. If the number of windows required to encode a complete message is greater than the size of this mask, 256 bits in this case, or 256 windows, then it simply resets the counter after window

255, and so on, until a complete message is encoded. It can then start over, or start on a new message.

The decoder has a bigger challenge to face. The decoder is given a set of masks, just like encoder. Unlike the encoder, the decoder cannot be sure that the first series of 128 samples it receives are the window 0 start of message, encoded by the decoder. The sample stream originally produced by an encoder may have been edited by clipping its ends randomly or splicing pieces together. In that case, the particular copy of the message that was clipped is unrecoverable. The decoder has the start of message delimeter used to encode the message that the decoder is looking for. In the initial state, the decoder assumes the first window it gets is window 0. It then decodes the proper number of bits dictated by the masks it was given. It compares these bits to the corresponding bits of the start of message delimeter. If they match, the decoder assumes it is still aligned, increments the window counter and continues. If the bits do not match, the decoder knows it is not aligned. In this case, it shifts one more sample onto the end of the sample buffer, discarding the first sample, and starts over. The window counter is set to 0. The decoder searches one sample at a time for an alignment lock. The decoder proceeds in this manner until it has decoded a complete match to the start of message delimeter or it exhausts the sample stream without decoding a message. If the decoder can match completely the start of message delimeter bit sequence, it switches into aligned mode. The decoder will now advance through the sample stream a full window at a time (128 samples). It proceeds until it has the 32 bits specifying the message size. This generally won't occupy more than 1 complete window. When the decoder has locked onto the start of message delimeter and decoded the message size, it can now proceed to decode as many consecutive additional windows as necessary until it has decoded a complete message. Once it has decoded a complete message, the state of the decoder can be reset to un-synchronized and the entire process can be repeated starting with the next 128 sample window. In this manner it is not absolutely necessary that encoding windows be contiguous in the sample stream. The decoder is capable of handling random intervals between the end of one message and the start of another.

It is important to note that the circuit for encoding and decoding a sample window does not need to be aware of the nature of the message, or of any structure beyond the start of message delimeter and message size. It only needs to consider a single sample window, its own state (whether the decoder is misaligned, synchronizing, or synchronized) and what bits to encode/decode.

Given that the stega-cipher apparatus allows for the encoding and decoding of arbitrary messages in this manner, how can it be used to protect copyrights?

The most important aspect of the stega-cipher in this respect is that fact that it makes the message integral with the content, and difficult to remove. So it cannot be eliminated simply by removing certain information prepended or appended to the sample stream itself. In fact, removing an arbitrary chunk of samples will not generally defeat the stega-cipher either.

Given that some information can be thus integrated with the content itself, the question is then how best to take advantage of this arrangement in order to protect copyrights.

The following protocol details how the stega-cipher will be exploited to protect copyrights in the digital domain.

In a transaction involving the transfer of digitized content, there are at least 3 functions involved:

The Authority is a trusted arbitrator between the two other functions listed below, representing parties who actually engage in the transfer of the content. The Authority maintains a database containing information on the particular piece of content itself and who the two parties engaged in transferring the content are. The Authority can perform stega-cipher encoding and decoding on content.

The Publisher, or online distributor is the entity which is sending the copyrighted content to another party. The Publisher can perform stega-cipher encoding and decoding on content.

The Consumer is the person or entity receiving the copyrighted content, generally in exchange for some consideration such as money. The consumer cannot generally perform stega-cipher encoding or decoding on content.

Each of these parties can participate in a message exchange protocol using well known public-key cryptographic techniques. For instance, a system licensing RSA public key algorithms might be used for signed and encrypted message exchange. This means that each party maintains a public key / private key pair, and that the public keys of each party are freely available to any other party. Generally, the Authority communicates via electronic links directly only to the Publisher and the Consumer communicates directly only with the publisher.

Below is an example of how the protocol operates from the time a piece of content enters an electronic distribution system to the time it is delivered to a Consumer.

A copyright holder (an independent artist, music publisher, movie studio, etc.) wishes to retail a particular title online. For instance, Sire Records Company might wish to distribute the latest single from Seal, one of their musical artists, online. Sire delivers a master copy of this single, "Prayer for the Dying", to the Authority, Ethical Inc. Ethical converts the title into a format suitable for electronic distribution. This may involve digitizing an analog recording. The title has now become content in the context of this online distribution system. The title is not yet available to anyone except Ethical Inc., and has not yet been encoded with the stega-cipher watermark. Ethical generates a Title Identification and Authentication

(TIA) certificate. The certificate could be in any format. In this example it is a short text file, readable with a small word-processing program, which contains information identifying

the title
the artist
the copyright holder
the body to which royalties should be paid
general terms for publishers' distribution
any other information helpful in identifying this content

Ethical then signs the TIA with its own private key, and encrypts the TIA certificate plus its signature with its own public key. Thus, the Ethical can decrypt the TIA certificate at a later time and know that it generated the message and that the contents of the message have not been changed since generation.

Sire Records, which ultimately controls distribution of the content, communicates to the Ethical a specific online Publisher that is to have the right of distribution of this content. For instance, Joe's Online Emporium. The Authority, Ethical Inc. can transmit a short agreement, the Distribution Agreement to the Publisher, Joe's Online Emporium which lists

the content title
the publisher's identification
the terms of distribution
any consideration paid for the right to distribute the content
a brief statement of agreement with all terms listed above

The Publisher receives this agreement, and signs it using its private key. Thus, any party with access to the Joe's Online Emporium's public key could verify that the Joe's signed the agreement, and that the agreement has not been changed since

Joe's signed it. The Publisher transmits the signed Distribution Agreement to the Authority, Ethical Inc.

Ethical Inc. now combines the signed TIA certificate and the Distribution Agreement into a single message, and signs the entire message using its private key. Ethical has now created a Publisher Identification message to go into its own stega-cipher channel in the content. Ethical Inc. now generates new stega-cipher masks and encodes this message into a copy of the content using a stega-cipher encoder. The Authority saves the masks as a Receipt in a database, along with information on the details of the transfer, including the title, artist and publisher.

Ethical then transfers this watermarked copy to the Joe's Online Emporium, the Publisher. Well known encryption methods could be used to protect the transfer between the Authority and the Publisher. The Authority may now destroy its copy, which the Publisher has received. The Publisher, Joe's Online Emporium now assumes responsibility for any copies made to its version of the content, which is a Publisher Master copy.

Finally, the Consumer, John Q. Public wishes to purchase a copy of the content from Joe's Online Emporium. Joe's Emporium sends the John Q. Public a short agreement via an electronic communication link, similar to Publisher's Distribution Agreement, only this is a Purchase Agreement, which lists

the content title
consumer identification
the terms of distribution
the consideration pas for the content
a brief statement of agreement with the terms above

John Q. Public signs this agreement with his private key and returns it to the Joe's Online Emporium. The Publisher, Joe's prepares to encode its own stega-cipher watermark onto a copy of the content by generating a set of masks for the algorithm. Joe's Online Emporium then stores these masks (a receipt) in its own database, indexed by title and consumer. Joe's Online Emporium signs the agreement received from John Q. Public with the Emporium's own private key, and forwards it to the Authority, Ethical Inc., along with a copy of the masks. It is important to note that this communication should be done over a secured channel. The Authority verifies the Publisher and Consumer information and adds its own signature to the end of the message, approving the transaction, creating a Contract of Sale. The Authority adds the Publisher's receipt (mask set) to its database, indexed by the title, the publisher, and the consumer identification. The Authority signs the Contract of Sale by encrypting it with their private key. So anyone with the Authority's public key (any Publisher) could decrypt the Contract of Sale and verify it, once it was extracted from the content. The Publisher then transmits the signed Contract of Sale back to the Publisher, who uses a stega-cipher device to imprint this Contract as its own watermark over the content. The Publisher then transmits the newly watermarked copy to the Consumer, who is accepting responsibility for it. The Publisher destroys their version of the consumer's copy.

If this procedure is followed for all content distribution within such an online system then it should be possible for the Authority to identify the owner of a piece of content which appears to be unauthorized. The Authority could simply try its database of stega-cipher keys to decode the watermark in the content in question. For instance, if a copy of Seal's latest single originally distributed with stega-cipher watermarks showed up on an Internet ftp site the Authority should be able to extract a TIA Certificate and Distribution Agreement or a Contract of Sale identifying the responsible party. If a Publisher sold this particular copy to a Consumer, that particular publisher should be able to extract a Contract of Sale, which places responsibility with the Consumer. This is not a time critical application, so even if it takes days or weeks, it is still worthwhile.

In a modification to the protocol discussed above, each Publisher might act as its own Authority. However, in the context of online services, this could open avenues of fraud committed by the collusion of certain Publishers and Consumers. Using an Authority, or one of several available Authorities to keep records of Publisher-Consumer transactions and verify their details decreases the likelihood of such events.

It should also be obvious that a similar watermarking system could be used by an individual entity to watermark its own content for its own purposes, wether online or in physical media. For instance, a CD manufacturer could incorporate unique stega-cipher watermarks into specific batches of its compact discs to identify the source of a pirate ring, or to identify unauthorized digital copies made from its discs. This is possible because the stega-cipher encoding works with the existing formats of digital samples and does not add any new structures to the sample data that cannot be handled on electronic or mechanical systems which predate the stega-cipher.

VI. Increasing Confidence in the Stega-Cipher

The addition of a special pre-encoding process can make stega-cipher certificates even more secure and undeniable. Hash values may be incorporated which match exactly the content containing the watermark to the message in the watermark itself. This allows us a verification that the watermark decoded was encoded by whomever signed it into this precise location in this specific content.

Suppose one wants to use a 256 bit (32 byte) hash value which is calculated with a secure one-way hash function over each sample in each sample window that will contain the message. The hash starts with a seed value, and each sample that would be processed by the encoder when encoding the message is incorporated into the hash as it is processed. The result is a 256 bit number one can be highly confident is unique, or sufficiently rare to make intentionally duplicating it with another series of samples difficult.

It is important that the hash function be insensitive to any changes in the samples induced by the stega-cipher itself. For instance, one might ignore the least significant bit of each sample when computing the hash function, if the stega-cipher was implemented using a least significant bit encode mode.

Based on the size of the non-hash message, one knows the hash-inclusive message requires 32 more bytes of space. One can now calculate the size of a signed encrypted copy of this message by signing and encrypting exactly as many random bytes as are in the message, and measuring the size of the output in bytes. One now knows the size of the message to be encoded. One can pre-process the sample stream as follows.

Proceed through the stega-cipher encode loop as described in the claims. Instead of encoding, however, calculate hash values for each window series which will contain the message, as each sample is processed. At the end of each instance of "encoding" take the resultant hash value and use it to create a unique copy of the message which includes the hash value particular to the series of sample windows that will be used to encode the message. Sign and encrypt this copy of the message, and save it for encoding in the same place in the sample stream.

A memory efficient version of this scheme could keep on hand the un-hashed message, and as it creates each new copy, back up in the sample stream to the first window in the series and actually encode each message, disposing of it afterwards.

The important result is evident on decoding. The decoding party can calculate the same hash used to encode the message for themselves, but on the encoded samples. If the value calculated by the decoding party does not match the value contained in the signed message, the decoder is alerted to the fact that this watermark was transplanted from somewhere else. This is possible only with a hash function which ignores the changes made by the stega-cipher after the hash in the watermark was generated.

This scheme makes it impossible to transplant watermarks, even with the keys to the stega-cipher.

Appendix - Psuedo-code

```
const int WINDOW_RESET = 256;
const int WINDOW_SIZE = 128;
const int MARKER_BITS = 1024;
const int CHUNK_BITS = 2048 * 8;

int window_offset;
int msg_bit_offset;
int frequency_offset;
Boolean useCell;

/* 8 bits per bye, 1 byte per char */
unsigned char frequency_mask[WINDOW_SIZE/8];
unsigned char window_mask[WINDOW_RESET/8];
unsigned char msg_start_marker[MARKER_BITS/8];
unsigned char msg_end_marker[MARKER_BITS/8];
Int16 amplitude_sample_buffer[WINDOW_SIZE];
float power_frequency_buffer[WINDOW_SIZE];
unsigned char message_buffer[CHUNK_BITS/8];

void doFFT(Int16 *amp_sample_buffer, float *power_freq_buffer,int size);
void doInverseFFT(Int16 *amp_sample_buffer, float *power_freq_buffer,int size);
void initialize();
Bit getBit(unsigned char *buffer,int bitOffset);
Boolean map(Bit window_bit, Bit band_bit, int window, int frequency);
Boolean getSamples(Int16 *amplitude_sample_buffer,int samples);
void encode()

void initialize()
{
        /* message to be encoded is generated */
        /* message is prefixed with 1024 bit msg_start_marker */
        /* message is suffixed with 1024 bit msg_end _marker */
        /* remaining space at end of message buffer padded with random bits */
        window_offset = 0;
        msg_bit_offset = 0;
        frequency_offset = 0;
        frequency_mask loaded
        window_mask loaded
        zeroAmpSampleBuffer();
}
```

```
Boolean getSamples(Int16 *buffer,int samples)
{
        /* get samples number of samples and shift them contiguously into the sample
          buffer from right to left*/
        if(samples < samples available)
                return false;
        else
                return true;
}

void doFFT(Int16 *sample_buffer, float *spectrum_buffer, int size)
{
        calculate FFT on sample_buffer, for size samples
        store result in spectrum buffer
}

void doInverseFFT(Int16 *sample_buffer,float *spectrum_buffer,int size)
{
        calculate inverse FFT on spectrum_buffer
        store result in sampe_buffer
}

Bit getBit(unsigned char *buffer,in bitOffset)
{
        returns value of specified bit in specified buffer
        either 0 or 1, could use Boolean (true/false) values for bit set of bit off
}

Boolean map(Bit window_bit,Bit band_bit,int window, int frequency_
{
        /* this is the function that makes the information difficult to find */
        /* the inputs window_bit and band_bit depend only on the mask values
                used for encoding the information, they are 1) random, 2) secret */
        /* window and frequency values are used add time and frequency band dependent
                complexity to this function */
        /* this function is equivalent to a Boolean truth table with window * frequency * 4
        possible input combinations and 2 possible output */
        /* for any input combination, the output is either true or false */
        /* window ranges from 0 to WINDOW_RESET -1 */
        /* frequency ranges from 0 to WINDOW_SIZE - 1 */
        return calculated truth value
}
```

```
void encodeBit(float *spectrum_buffer,int freq_offset,Bit theBit)
{
        /* modifies the value of the cell in spectrum_buffer, indexed by freq_offset
                in a manner that distinguishes each of the 2 possible values of theBit,
                1 or 0
        */
        /* suggested method of setting the Least Significant bit of the cell == theBit */
        /* alternative method of rounding the value of the cell upward or downward to
                certain fractional values proposed
                i.e. <= .5 fractional remainder signifies 0, > .5 fraction remainder
                        signifies 1
        */
}

void encode()
{
        initialize();


        do {

        if(getSamples(amplitude_sample_buffer) == false)
                return

        doFFT(amplitude_sample_buffer,power_frequency_buffer,WINDOW_SIZE);

        for (frequency_offset = 0; frequency_offset < WINDOW_SIZE;
        frequency_offset++){


                useCell = map(getBit(window_mask,window_offset),
                                getBit(frequency_mask,frequency_offset),
                                window_offset, frequency_offset);

                if(useCell == true){
                        encodeBit(power_frequency_buffer,frequency_offset,
                                getBit(message_buffer,msg_bit_offset));
                        message_bit_offset ++;
                        if(msg_bit_offset == MESSAGEBITS){
                                initialize();
                                break; /* exit frequency loop */
                        }
                }
        }
```

```
        doInverseFFT(amplitude_sample_buffer,power_frequency_buffer,
                WINDOW_SIZE);

        outputSamples(amplitude_sample_buffer);

        window_offset++;
        if(window_offset == WINDOW_RESET){
                window_offset = 0;
        }


        }while(true);
}
```

The encode() procedure processes an input sample stream using the specified frequency and window masks as well as a pre-formatted message to encode.

encode() processes the sample stream in windows of WINDOW_SIZE samples, contiguously distributed in the sample stream, so it advances WINDOW_SIZE samples at a time.

For each sample window, encode() first compute the FFT of the window, yielding its Power Spectrum Estimation. For each of these window PSEs, encode() then uses the map() function to determine where in each PSE to encode the bits of the message, which it reads from the message buffer, on ebit at a time. Each time map() returns true, encode() consumes another sample from the message.

After each window is encoded, encode() computes the inverse FFT on the PSE to generate a modified sample window, which is then output as the modified signal. It is important the sample windows NOT overlap in the sample stream, since this would potentially damage the preceeding encoding windows in the stream.

Once the message is entirely encoded, including its special end of message marker bit stream, encode() resets it internal variables to begin encoding the message once more in the next window. encode() proceeds in this manner until the input sample stream is exhausted.

```
enum {
        Synchronizing,
        Locked
}; /* decode states */
```

```
unsigned char message_end_buffer[MARKER_BITS];

Bit decodeBit(float *spectrum_buffer,int freq_offset)
{
        /* reads the value of the cell in spectrum_buffer, indexed by freq_offset
                in a manner that distinguishes each of the 2 possible values of an
                encoded bit, 1 or 0
        */
        /* suggested method of testing the Least Significant bit of the cell */
        /* alternative method of checking the value of the cell versus certain fractional
remainders proposed.
                i.e. <= .5 fractional remainder signifies 0, > .5 fraction remainder
                        signifies 1
        */
        return either 1 or 0 as appropriate
}

Boolean decode()
{
        /* Initialization */
        state = Synchronizing
        window_offset = 0;
        set frequency mask
        set window mask
        clear sample buffer
        int nextSamples = 1;
        int msg_start_offset = 0;
        clear message_end_buffer
        Bit aBit;
        Boolean bitsEqual;

        do {

                if(state == Synchronizing){
                        nextSamples = 1;
                        window_offset = 0;
                }
                else
                        nextSamples = WINDOW_SIZE;

                if(getSamples(amplitude_sample_buffer) == false)
                        return false;
```

```
                doFFT(amplitude_sample_buffer,power_frequency_buffer,
                        WINDOW_SIZE);  /* 2 */

                for (frequency_offset = 0; frequency_offset < WINDOW_SIZE;
                frequency_offset++){

                        useCell = map(getBit(window_mask,window_offset),
                                getBit(frequency_mask,frequency_offset),
                                window_offset, frequency_offset);

                        if(useCell == true){
                                aBit = decodeBit(power_frequency_buffer,
                                                frequency_offset);
                                setBit(message_buffer,message_bit_offset,aBit);
                                message_bit_offset ++;
                        }
                        else
                                continue;
                        if(state == Synchronizing){
                                bitsEqual =
                                compareBits(message_start_marker,message_buffer,
                                        message_bit_offset);
                                if(!bitsEqual){
                                        message_bit_offset = 0;
                                        misaligned = true;
                                        break; /* exit frequency loop */
                                }
                                else if (message_bit_offset == MARKER_BITS)
                                        state == Locked;
                        }
                        else {
                                /* locked onto encoded stream */
                                shift aBit into right side of message_end_buffer
                                bitsEqual = compareBits(message_end_buffer,
                                        msg_end_marker,MARKER_BITS);
                                if(bitsEqual)
                                        return true;
                        }
                }

        }while (true);
}
```

The decode() procedure scans an input sample stream using specified window and frequency masks, until it either decodes a valid message block, storing it in a message buffer, or exhausts the sample stream.

The decode() procedure starts in state Synchronizing, in which it does not know where in the sample stream the encoding windows are aligned. The procedure advances the sample window through the sample stream one sample at a time, performing the FFT calculation on each window, and attempting to decode valid message bits from the window. As it extracts each bit using the map() function, the decode() procdure compares these bits against the start of message marker. As soon as a mismatch is detected, the decode() procedure knows it is not yet properly aligned to an encoding window, and immediately ceases decoding bits from the current window and moves to the next window, offset by 1 sample. The decode() procedure continues in this manner until it matches successfully the complete bitstream of a start of message marker. At this point the decode() procedure assumes it is aligned to an encoded message and can then decode bits to the message buffer quickly, advancing the sample window fully at each iterations. It is now in Locked mode. For each bit it stores in the message buffer when in Locked mode, the decode() procedure also shifts the same bit value into the least significant bit of the message_end_buffer. After each bit is decoded in Locked mode, the decode() procedure checks compares the message_end_buffer with the msg_end_marker in a bit by bit manner. When a complete match is found, decode() is finished and returns true. If the sample stream is exhausted before this occurs, decode() returns false. If decode() returns true, a valid message is stored in the message buffer, including the start and end of message markers.

Claims

1. A steganographic method comprising the steps of :

using random keys in combination with steganography to encode additional information into digitized samples such that a signal generated from the modified sample stream is not significantly degraded and such that the additional information cannot be extracted without the keys and such that the signal generated from the modified sample stream will be degraded by attempts to erase, scramble, or otherwise obliterate the encoded additional information.

2. An apparatus for encoding or decoding a message, represented as series of data bits into or out of a series of digitized samples, comprising:

a) a sample buffer for holding and accessing and transforming digitized samples;

b) a digital signal processor capable of performing fast fourier transforms;

c) a memory to contain information representing

1) primary mask,

2) convolutional mask,

3) start to message delimiter,

4) a mask calculation buffer,

5) a message buffer,

6) an integer representing a message bit index,

7) a position integer M representing message size,

8) an integer representing an index into said primary mask,

9) an integer representing an index into said convolution mask,

10) an integer representing the state of a decode process,

11) a table representing a map function;

12) a flag indicating a complete message has been decoded or encoded,

13) a positive integer S representing a number of samples to read into said sample buffer, and

14) a flag indicating the size of a message which has been decoded;

d) an input to acquire digital samples;

e) an output to output modified digital samples;

f) an input for inputting the values of (c1) - (c5) and (c11) and (c13);

g) an output to output the message stored in (c5) as the result of a decode process and the value of (c10) to an attached digital circuit;

h) at least one data bus to transfer information from

(d) to (a),

(a) to (b),

(b) to (a),

(a) to (e),

(f) to (c), and

(c) to (e); and

i) a clock which generates a clock signal to drive (b) and control the operation of the apparatus.

3. A method of encoding information into a sample stream of data, said method comprising the steps of:

A) generating a mask set to be used for encoding, said set including:

a random or pseudo-random primary mask,

a random or pseudo-random convolution mask,

a random or pseudo-random start of message delimiter, wherein said mask set can be concatenated and manipulated as a single bit stream;

B) obtaining a message to be encoded;

C) generating a message bit steam to be encoded such that the stream includes

1) a start of message delimiter, and

2) an integer representing the number of message bytes to follow the message;

D) loading the message bit stream, a map table, the primary mask, the convolution mask, and the start of message delimiter into a memory;

E) resetting a primary mask index, a convolution mask and message bit index, and setting the message size integer equal to the total number of bits in the message bit stream;

F) clearing a message encoded flag;

G) reading a window of samples from a sample input device and storing them sequentially in a sample buffer;

H) resetting the primary mask index and looping through the sample buffer from a first sample to a last sample incrementing the primary mask index each time a sample is visited, such that for each sample position, a value of the mapping function is computed, which is either true or false, by using a bit of the primary mask representing a current sample and a bit of the convolution mask indicated by the convolution index to calculate an offset in the map table;

I) obtaining the bit value stored in the map table and encoding the bit of the message indicated by the message bit index into the current sample if the bit value obtained from the map table is a certain value and incrementing the message bit index, determining whether the message bit index equals the number of message bits, and if it does re-performing step A), setting the message encoded flag, and exiting the loop;

J) outputting the modified samples in the sample buffer, and if the message encoded flag is set jumping back to said step E);

K) incrementing the convolution index, wherein if the convolution index equals the length of the convolution mask in bits then set the convolution index to 0; and

L) jumping back to step G).

4. A method of encoding information into a sample stream of data, comprising the steps of:

A) generating a mask set to be used for encoding, including:

a random or pseudo-random primary mask,

a random or pseudo-random convolution mask, and

a random or pseudo-random start of message delimiter, wherein said mask set can be concatenated and manipulated as a single bit stream;

B) inputting a message to be encoded;

C) generating a message bit stream to be encoded including

a start of message delimiter, and

an integer representing of number of message bytes to follow the message;

D) loading the message bit stream, a map table, and the mask set into a memory;

E) resetting a primary mask index, a convolution mask and message bit index, setting the message size index equal to the number of bits in the message bitstream, and clearing a message encoded flag;

F) reading a window of samples of the inputted message and storing the samples sequentially in a sample buffer;

G) computing a spectral transform of the samples in the buffer;

H) obtaining the bit value stored in the map table, wherein if the bit value is true, then encoding the bit of the message indicated by the message bit index into the current sample and incrementing the message bit index, where the message bit index equals the number of message bits, and then reperforming step A), setting the message encoded flag, and exiting the loop;

I) computing the inverse spectral of the spectral values stored in the sample buffer;

J) outputting the values in the sample buffer, and if the sample encoded flag is set, then clear the flag and jump back to step E);

K) incrementing the convolution index and when the convolution index equals the length of the convolution mask in bits resetting the convolution index; and

L) jumping back to step F).

5. The method of claim 3 wherein the encoding of the message bit into the sample in step I includes encoding a single bit of the sample to match the message bit.

6. The method of claim 4 wherein the encoding of the message bit into the sample in step H includes altering the sample value such that said sample value falls within a prespecified range of valves relative to its original value.

7. A method of decoding information from a sample stream of data, comprising the steps of:

A) obtaining a mask set including:

(1) a random or pseudo-random primary mask,

(2) a random or pseudo-random convolution mask, and

(3) a random or pseudo-random start of message delimiter;

B) loading a map table, and the mask set into a memory;

C) resetting a primary mask index and convolution mask index and setting a message size integer equal to 0;

D) clearing a message decoded flag;

E) setting a state of the decode process to SYNCHRONIZED;

F) checking the state of the decode process and if the decode state is UNSYNCHRONIZED, setting a number of samples to equal 1 and resetting the convolution index to 0; otherwise, setting the number of samples to equal S ($S \geq 1$);

G) reading the number of samples specified in step F) into a sample buffer;

H) resetting the primary mask index, and looping through the sample buffer from the first sample to the last sample, incrementing the primary mask index each time, and for each sample position, computing the value of a mapping function to calculate an offset into the map table;

I) obtaining the bit value in the map table, and if the value is true, decoding the bit of the message indicated by the message bit index, storing the bit into the message buffer at the message bit index, and incrementing the message bit index;

J) comparing the decoded bits in the message buffer to the start of message delimiter, and if the number of bits in the message buffer is less than or equal to the number of bits in the start of message delimiter and the bits match, then setting the state of the decode process to SYNCHRONIZED; otherwise setting the state of the decode process to UNSYNCHRONIZED;

K) if the state of the decode process is SYNCHRONIZED and the number of bits in the message buffer is greater than or equal to the sum of the number of bits of the start of delimiter and the message size, then setting the state of the decode process to SYNC-AND-SIZE and copying certain bits from the message buffer to a message size integer container;

L) if the state of the decode process is SYNC-AND-SIZE and the number of bits in the message buffer divided by 8 is greater than or equal to the message size, then setting the message decoded flag, outputting the message and the message decoded flag and ending the method;

M) incrementing the convolution index, and if the convolution index equals the number of bits in the convolution mask resetting the convolution index; and

N) jumping to step F).

8. A method of decoding information from sampled data, comprising the steps of:

A) Obtaining a mask set including

(1) a random or pseudo-random primary mask,

(2) a random or pseudo-random convolution mask, and

(3) a random or pseudo-random start of message delimiter;

B) loading a map table, and the mask set into a memory;

C) resetting a primary mask index and convolution mask index and setting a message size integer equal to 0;

D) clearing a message decoded flag;

E) setting a state of the decode process to SYNCHRONIZED;

F) checking the state of the decode process and if the decode state is UNSYNCHRONIZED, setting a number of samples to equal 1 and resetting the convolution index to 0; otherwise, setting the number of samples to equal S (S>1);

G) reading the number of samples specified in step F) into a sample buffer;

H) computing a spectral transform of the samples stored in the sample buffer;

I) resetting the primary mask index and looping through the sample buffer from the first sample to the last sample, incrementing the primary mask index each time, and for each sample position, computing the value of a mapping function by using the bit of the primary mask corresponding to the primary mask index and the bit of the convolution masks indicated by the convolution phase to calculate an offset into the map table representing the mapping function;

J) obtaining a bit value stored in the map, and if the value is true, decoding the bit of the message indicated by the message bit index from the current sample, storing the bit into the message buffer at the message bit index, and incrementing the message bit index;

K) comparing the decoded bits in the message buffer to the start of message delimiter, and if the number of bits in the message buffer is less than or equal to the number of bits in the start of message delimiter and the bits match, then setting the state of the decode process to SYNCHRONIZED; otherwise, setting the state of the decode process UNSYNCHRONIZED;

L) if the state of the decode process is SYNCHRONIZED, and the number of bits in the message buffer is greater than or equal to the sum of the number of bits of the start of delimiter and the message size, then setting the state of the decode process to SYNC-AND-SIZE and copying certain bits from the message buffer to a message size integer container;

M) if the state of the decode process is SYNC-AND-SIZE and the number of bits in the message buffer divided by 8 is greater than or equal to the message size, then setting the message decoded flag, outputting the message and the message decoded flag and ending the method;

N) incrementing the convolution index, wherein if the convolution index equals the number of bits in the convolution mask, then resetting the convolution index; and

O) jumping to step F).

9. The method of claim 7 wherein the decoding of the message bit from the sample in step I includes reading a single bit of the sample.

10. The method of claim 7 wherein the decoding of the message bit from the sample in step I includes mapping a range of sample values onto a particular message bit value.

11. The method of claim 4 wherein the map table is defined such that any index of the map table directs the process to encode information.

12. The method of claim 1 wherein the samples are obtained from a sample stream representing digitized sound or music.

13. The method of claim 12 wherein the identical encode process is performed on two sample streams representing channel A and channel B of digitized stereo sound.

14. The method of claim 12 wherein the sample streams represent channel A and channel B of digitized stereo sound and are interleaved before being input as a single sample stream and are separated into two channels upon output.

15. The method of claim 1 wherein the samples are obtained from a sample stream representing digitized video.

16. The method of claim 1 wherein the samples are obtained from a sample stream representing a digitized image.

17. The apparatus of claim 2, further comprising a tamper-resistant packaging, enclosing said apparatus wherein circuitry and information stored therein are destroyed if said packaging is opened.

18. The method of claim 3, further comprising a pre-encoding step which customizes the message to be encoded including: calculating over which windows in the samples stream a message will be encoded, computing a secure one way hash function of the samples in those windows, and placing the resulting hash values in the message before the message is encoded;

wherein the hash calculating step includes: calculating the size of the original message plus the size of an added hash value, and pre-processing the sample stream for the purpose of calculating hash values of each series of windows that will be used to encode the message and creating a modified copy of the message containing the hash value such that each message containing a hash value matches each window series uniquely.

19. The method of claim 1, wherein an authority for on line distribution of content encodes at least one of the following items into a sample stream ;

the title,

the artist,

the copyright holder,

the body to which royalties should be paid, and

general terms for publisher distribution.

20. The method of claim 19, wherein the authority combines at least one item with a secure private key signed message from a publisher containing at least one of the following pieces of information:

the title,

the publisher's identification,

the terms of distribution,

any consideration paid for the right to distribute the content,

a brief statement of agreement, and

the publisher signs and encrypts the combined message using a public key cryptosystem and encodes the signed and encrypted message into the sample stream.

21. The method of claim 20, wherein a publisher obtains the encoded sample stream and additionally obtains information form the authority and combines this with a message received from a consumer, which has been signed using a public key cryptosystem and wherein the signed message contains at least one of the following data

the content title,

consumer identification,

the terms of distribution,

the consideration paid for the content,

a brief statement of agreement, and

the publisher uses a public key cryptosystem to sign the combined information and finally encodes the signed information.

22. The method of claim 1, wherein the sample stream is obtained from at least one audio track contained within a digitized movie, video game software, or other software.

23. The method of claim 1, wherein the sample stream is obtained from at least one digitized movie or still image contained within a video game or other software.

24. The method of claim 1, wherein encoded information is contained in the differences or relationship between samples or groups of samples.

25. The method of claim 4, wherein the encoding of the message bit into the sample in step H includes encoding a single bit of the sample to match the message bit.

26. The method of claim 3, wherein the encoding of the message bit into the sample in step I includes altering the sample value such that said sample value falls within a prespecified range of valves relative to its original value.

27. The method of claim 8, wherein the decoding of the message bit in step J includes reading a single bit of the sample.

28. The method of claim 8, wherein the decoding of the message bit in step J includes mapping a range of supply values onto a particular message bit value.

29. The method of claim 3, wherein the map table is defined such that any index of the map table directs the process to encode information.

30. The method of claim 7, wherein the map table is defined such that any index of the map table directs the process to encode information.

31. The method of claim 8, wherein the map table is defined such that any index of the map table directs the process to encode information.

32. The method of claim 4, further comprising a pre-encoding step which customizes the message to be encoded including: calculating over which windows in the samples stream a message will be encoded, computing a secure one way hash function of the samples in those windows, and placing the resulting hash values in the message before the message is encoded;

wherein the hash calculating step includes: calculating the size of the original message plus the size of an added hash value, and pre-processing the sample stream for the purpose of calculating hash values of each series of windows that will be used to encode the message and creating a modified copy of the message containing the hash value such that each message containing a hash value matches each window series uniquely.--