**PCT** WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

**(51) International Patent Classification** 6 :
**H04L 9/00** | **A1**

**(11) International Publication Number:** **WO 00/54453**

**(43) International Publication Date:** 14 September 2000 (14.09.00)

**(21) International Application Number:** PCT/US00/06296

**(22) International Filing Date:** 10 March 2000 (10.03.00)

**(30) Priority Data:**

| | | |
|---|---|---|
| 60/123,581 | 10 March 1999 (10.03.99) | US |
| 60/123,587 | 10 March 1999 (10.03.99) | US |
| 60/126,591 | 26 March 1999 (26.03.99) | US |
| 60/126,592 | 26 March 1999 (26.03.99) | US |
| 09/404,292 | 23 September 1999 (23.09.99) | US |
| 09/404,291 | 23 September 1999 (23.09.99) | US |

**(71) Applicants** *(for all designated States except US)*: DIGIMARC CORPORATION [US/US]; 19801 SW 72nd Avenue, Suite 250, Tualatin, OR 97062 (US). ACOUSTIC INFORMATION PROCESSING LAB, LLC. [US/US]; 110 NE Cedar Street, Stevenson, WA 98648 (US).

**(72) Inventor; and**
**(75) Inventor/Applicant** *(for US only)*: LEVY, Kenneth, L. [US/US]; 110 NE Cedar Street, Stevenson, WA 98648 (US).

**(74) Agent:** CONWELL, William, Y.; Digimarc Corporation, 19801 SW 72nd Avenue, Suite 250, Tualatin, OR 97062 (US).

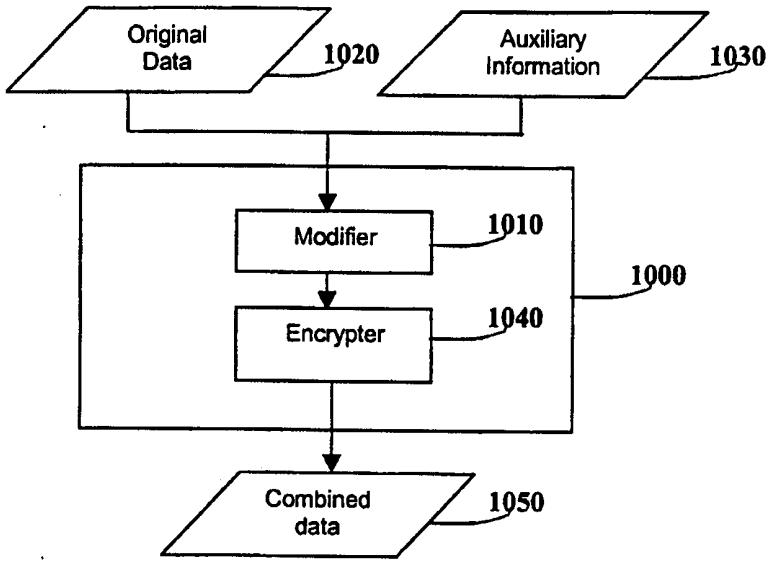
**(81) Designated States:** AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

**Published**
*With international search report.*
*Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.*

**(54) Title:** SIGNAL PROCESSING METHODS, DEVICES, AND APPLICATIONS FOR DIGITAL RIGHTS MANAGEMENT

**(57) Abstract**

Techniques are detailed for steganographically embedding auxiliary data (1030) within electronic content (e.g., audio, video, still imagery, etc.) (1020) in manners that are computationally simple, yet highly inconspicuous (1010). The embedded data can convey copyright or other ownership information, or may be used for device control purposes (e.g., preventing unauthorized reproduction). A number of countermeasures against removal of the auxiliary data are contemplated, including keying use of the content to the presence of such data. The embedded data may be made dependent on the media encoded, e.g., by modifying the embedded data in accordance with characteristics from the media. Encryption can also advantageously be employed (1040). Playback devices may be equipped to track IDs from previously–accessed content, and enforce usage rules. Some embodiments employ multiple watermarks to advantage, e.g., a robust watermark is encoded prior to distribution and indicates the content is protected, and a second watermark is encoded by the playback device and serves to uniquely link that content to that device. Some applications benefit from scrambling of content, in a manner that leaves certain information (e.g., from a header) unscrambled and freely accessible.



*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.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **AL** | Albania | **ES** | Spain | **LS** | Lesotho | **SI** | Slovenia |
| **AM** | Armenia | **FI** | Finland | **LT** | Lithuania | **SK** | Slovakia |
| **AT** | Austria | **FR** | France | **LU** | Luxembourg | **SN** | Senegal |
| **AU** | Australia | **GA** | Gabon | **LV** | Latvia | **SZ** | Swaziland |
| **AZ** | Azerbaijan | **GB** | United Kingdom | **MC** | Monaco | **TD** | Chad |
| **BA** | Bosnia and Herzegovina | **GE** | Georgia | **MD** | Republic of Moldova | **TG** | Togo |
| **BB** | Barbados | **GH** | Ghana | **MG** | Madagascar | **TJ** | Tajikistan |
| **BE** | Belgium | **GN** | Guinea | **MK** | The former Yugoslav Republic of Macedonia | **TM** | Turkmenistan |
| **BF** | Burkina Faso | **GR** | Greece | | | **TR** | Turkey |
| **BG** | Bulgaria | **HU** | Hungary | **ML** | Mali | **TT** | Trinidad and Tobago |
| **BJ** | Benin | **IE** | Ireland | **MN** | Mongolia | **UA** | Ukraine |
| **BR** | Brazil | **IL** | Israel | **MR** | Mauritania | **UG** | Uganda |
| **BY** | Belarus | **IS** | Iceland | **MW** | Malawi | **US** | United States of America |
| **CA** | Canada | **IT** | Italy | **MX** | Mexico | **UZ** | Uzbekistan |
| **CF** | Central African Republic | **JP** | Japan | **NE** | Niger | **VN** | Viet Nam |
| **CG** | Congo | **KE** | Kenya | **NL** | Netherlands | **YU** | Yugoslavia |
| **CH** | Switzerland | **KG** | Kyrgyzstan | **NO** | Norway | **ZW** | Zimbabwe |
| **CI** | Côte d'Ivoire | **KP** | Democratic People's Republic of Korea | **NZ** | New Zealand | | |
| **CM** | Cameroon | | | **PL** | Poland | | |
| **CN** | China | **KR** | Republic of Korea | **PT** | Portugal | | |
| **CU** | Cuba | **KZ** | Kazakstan | **RO** | Romania | | |
| **CZ** | Czech Republic | **LC** | Saint Lucia | **RU** | Russian Federation | | |
| **DE** | Germany | **LI** | Liechtenstein | **SD** | Sudan | | |
| **DK** | Denmark | **LK** | Sri Lanka | **SE** | Sweden | | |
| **EE** | Estonia | **LR** | Liberia | **SG** | Singapore | | |

# SIGNAL PROCESSING METHODS, DEVICES, AND APPLICATIONS FOR DIGITAL RIGHTS MANAGEMENT

## Field of the Invention

This invention relates to the field of signal processing, and more particularly relates to techniques useful in encoding audio, video, and other content for digital rights management purposes.

## Background of the Invention

With the recent explosive growth in the use of electronic information, enforcement of copyright laws has become more difficult. The cost of the equipment required to copy digital data representing music, art, and other valuable information has been decreasing, while the capacity of readily available data storage media has been increasing. Inexpensive devices can write enormous amounts of data to digital storage media such as writable compact disks (CD-R or CD-RWs), multi-gigabyte hard disk drives, high capacity removable magnetic disks, and soon to be available digital versatile disks (DVDs). Readily available high-resolution printers and scanners bring the digitization and reproduction of graphic information within the means of most consumers. In addition, readily available high-resolution sound cards, including analog-to-digital and digital-to-analog converters, bring the digitization and reproduction of audio information within the means of most consumers. Not only is copying digital files simple and inexpensive, the Internet facilitates unauthorized distribution of copyrighted works.

Unlike analog copies, which are always inferior to the original, a copy of digital information can be identical to that of the original, with no degradation due to copying. Millions of dollars are lost annually due to illegal but exact duplications of digital media and near-exact duplications of analog media. Because copying equipment is readily available, catching persons making unauthorized copies can be difficult. Even if an unauthorized copier is apprehended, the creator of the original document must still prove that the allegedly unauthorized copy was in fact copied from his original work and not independently created.

In one aspect, the technology detailed below relates to digital watermarking, or "data hiding," and its use in solving the problem of illegal copying (e.g., hiding authentication information or copy protection information within the original data). Hiding auxiliary information in original data, also called steganography, has been used for thousands of years. In steganography, a message is hidden within another object or media, so that the message is essentially imperceptible to a human observer (or listener). Steganography is related to, but different from, cryptography, in which the existence of a message is typically obvious, but its meaning is not ascertainable without special knowledge.

Hidden data, also referred to as auxiliary or embedded data, can be used to prevent unauthorized copying by embedding in the original data commands that are readable by the copying device and that instruct the copying device not to make a usable copy. Hidden data can also be used to authenticate data, that is, to prove authorship. One such technique entails embedding auxiliary information in an original work in such a manner that special knowledge, such as a secret algorithm or code, is required to detect

and/or remove the auxiliary information. The copier would not be able to remove the authentication information, and the original creator could prove his authorship by retrieving the embedded information, which would identify him as the author.

Data hiding has uses besides the prevention and detection of unauthorized copying. One such use is content enhancement, that is, adding information to the original data to enhance the content. For example, lyrics could be embedded in audio data on a CD. The lyrics could be viewed in a special karaoke machine, while the audio could be played on an existing CD player. Hidden data could also be used to associate different segments of video data with different viewer-selectable versions of the video on a DVD. For example, a viewer could select between a version edited for children or an unabridged version, and embedded auxiliary data would indicate to the DVD player which video segments to skip and which to include for the selected version.

The original data in which the auxiliary data is hidden may represent any type of information that is perceivable with the aid of a presenting device. For example, the data may represent music which is presented using a compact disk or audio DVD player, a video film that is presented on a DVD player, or an image that is presented on a computer screen or a printer.

When the combined data is presented to a user by a normal presentation device, the auxiliary data should not interfere with the use of the original data. Ideally, the user should not be able to perceive the auxiliary data at all. Unfortunately, increasing the amount of the embedded auxiliary data or its robustness, that is, its persistence to attack and data transformation, may incidentally increase its perceptibility. The degree to which the auxiliary data can be perceived without having an adverse impact on the user varies with the application. For example, in CD quality audio, a minor change from the original data might result in unacceptable audio artifacts. In video data, a minor change in a presented image may be acceptable, even though the change might be noticeable if the original and combined works are presented and compared side by side.

Several techniques are known for hiding auxiliary information in original digital data. Data can be hidden in original data as headers or trailers appended to the original data. Such techniques are of limited use in protection of copyrighted works, because the auxiliary data is easily located and stripped out of the copy, as when changing format. More sophisticated techniques distribute the auxiliary data through the original data, entwining the auxiliary and original data until the auxiliary data is difficult, or even statistically impossible, to identify and strip from the combined data.

Most data hiding techniques that distribute the auxiliary data through the original data are computationally intense and therefore expensive to implement. Many of these techniques are based upon adding or subtracting periods of pseudo-random noise (PN) sequences with the signal to represent the auxiliary information, and these sequences may require filtering (a.k.a. shaping) in the frequency domain. The rest are based upon adding the auxiliary information to the original data after the original data has been transformed into the frequency domain, such as by a Fourier transform. Auxiliary information can be added in the frequency domain so that the energy of the auxiliary data is spread across many frequencies in a manner similar to that of the PN sequence. In addition, auxiliary information can be

added to the phase of the frequency components with and without spreading the information across frequencies. Unfortunately, transforming the data into the frequency domain and/or shaping the energy of the PN sequence so it is less perceivable requires intense calculations.

The ability of users to detect auxiliary data depends not only upon the data, but also upon the characteristics of the human sense organs and the interpretation of sensory stimuli by the brain. Some data hiding techniques transform the original data into the frequency domain and embed auxiliary data in a manner such that the frequency spectrum of the original data reduces the perception of embedded data. This psychophysical effect is known as masking. The frequency distribution of the original data is used to determine preferred frequencies at which the embedded auxiliary data will be less perceptible, that is, masked. Others use the fact that we don't perceive phase as accurately as magnitude in the frequency domain.

There are some data embedding techniques that are less computationally intense and that still distribute the auxiliary data in the original data. Such techniques include amplitude modulation, frequency band elimination, distinct quantization, and least-significant bit (LSB) replacement. These techniques embed data in predetermined locations without regard to the original data and are, therefore, more likely to produce perceptual side affects in the combined data. In addition, the LSB replacement technique is easily disturbed by low level noise.

The ease of retrieving embedded data varies with the technique used for embedding. Some data hiding and retrieving techniques retrieve the auxiliary data by comparing the combined data with the original data. Others retrieve the auxiliary information using databases of the PN sequences that were originally used to hide the data. Techniques that require that a copy of the original data or a PN database be used to extract the auxiliary data are of limited use in applications in which the combined data is distributed broadly. Such techniques are useful in some applications, such as data authentication, in which the auxiliary data is retrieved rarely and only by the copyright owner.

Embedded data techniques are susceptible to removal of auxiliary information for either of the following reasons. First, the very nature of embedded data is incompatible with bit-rate reducing (a.k.a. compression) schemes, which remove the non-perceivable aspects of the data such as done with MPEG compression. Since a key feature of any embedded data is the fact that it is non-perceivable, compression schemes will act to remove the embedded data. Even if the embedded data is designed to survive the current compression technology, the next generation technology will probably remove it. Bit-rate compression schemes are very important in the digital distribution of media, and receiving much research. Second, noise reduction techniques will be able to remove embedded data. Noise reduction techniques are a hot topic, and used to restore old recordings. Since most non-perceivable embedded data is similar to noise, it will be removed by these noise reduction techniques. Again, even if the embedded data is designed to survive the current restoration technology, the next generation technology may probably remove it.

In another aspect, the technology detailed herein relates to methods and systems for making embedded data more robust against corruption and attack.

Embedded data may be susceptible to various types of corruption and attack. For example, the very nature of embedded data technology is at odds with bit-rate reducing (a.k.a. compression) schemes, which remove the non-perceivable aspects of the data such as done with MPEG compression. Since a feature of most embedded data is that it is non-perceivable, compression schemes will tend to remove the embedded data. Even if the embedded data is designed to survive the current compression technology, the next generation technology may result in its removal. Bit-rate compression schemes are very important in the digital distribution of media, and receiving much research. Likewise, noise reduction techniques, e.g., as are used to restore old audio recordings, pose a threat to embedded data. Since most non-perceivable embedded data is similar to noise, it will be removed by these noise reduction techniques. Again, even if the embedded data is designed to survive the current restoration technology, the next generation technology will probably remove it.

In another aspect, the technology detailed herein relates to ID assignment and binding. Content providers may want to allow only the person who bought content to access (i.e. play, copy or record) that content. One way to do this is to provide content that contains an ID, and lock the ID to the consumer, the rendering device or the storage unit. However, these existing solutions of how to use the ID produce unreasonable burdens for consumers.

One existing solution, known as user-binding, requires a person to carry an ID-card and/or remember a personal identification number (PIN) to access the content, similar to the way bank ATM machines work. The consumer has accepted this solution in order to access money in the bank, a situation where security is an advantage to the consumer too. However, it is doubtful that consumers will accept this requirement to access content, for example, play audio on a car stereo. In addition, when a group of people are sharing content, such as music, the process of each person having to scan a card before listening to their music is obtrusive. Finally, this solution requires data that links the ID to the user, so PINs and/or ID-cards can be produced. This data means the user's privacy has been compromised.

Another existing solution restricts playing of the content to one device, known as player-binding. This solution means your friend's music will not play in your car stereo, neither will your movie play at his house. This solution is not only inconvenient to the consumer, but also reduces the sale of content since many people buy content after playing or viewing it with their friends.

A final solution links the content to the storage unit, known as media-binding. The storage unit includes but is not limited to a magnetic hard drive, optical disk or electronic memory. This solution becomes cumbersome when the content should be allowed to move between different storage unit types. For example, a user, Joe, may want to play his audio from his computer's hard drive over his home stereo, or have the audio in his car or on a jog as portable electronic memory. However, with this media-binding solution, this audio can only be played in one place, and to move it from Joe's stereo to his car, he has to remember to where it was "checked out", otherwise, piracy cannot be controlled. Importantly, he can't just listen to it from each place as desirable to the consumer.

Another aspect of the technology detailed herein involves the use of multiple embedded data – with different characteristics – to serve rights management functions.

Another aspect of the technology detailed herein relates to scrambling of content to protect same.

It is often desirable to degrade digital signals so as to restrict access. For instance, pay-TV broadcasts are degraded so those who haven't paid for the program cannot watch it because the picture is unclear, while those who have paid for the program see a clear picture because their recovery apparatus has been enabled. Most recently, as a result of the digital audio revolution, it is desirable to restrict MP3 (a standard bit-rate compressed audio file format) access. It is also desirable to produce inexpensive portable MP3 players, which in turn require that recovery of the original signal be simple.

There are numerous existing methods of degrading digital content, a.k.a. scrambling. Some methods require a key to de-scramble the content, whereas others do not. Most scrambling or degrading methods are based upon either adding an interference signal to the digital content, or moving the bits around. Other methods use encryption, but this is very computationally intense.

It would be advantageous if the information about a movie on a scrambled channel could be displayed to a viewer without having to de-scramble the information.

More recently, as a result of the digital audio revolution, some people would like to see MP3 (a Motion Pictures Expert Group Layer III standard bit-rate reduced audio file format) access restricted as it is easy to make an exact digital copy of the content. This restriction can be implemented via scrambling techniques. However, it is desirable to retrieve information about the scrambled song without de-scrambling the information, as this would allow a user to learn about a song before deciding whether or not to play that song, thus improving speed of the system for the user. It would also allow a user's player to quickly read copyright information, which could then enable playing. It is also desirable to produce inexpensive portable MP3 players, which in turn require that recovery of the original signal be simple.

The prior-art contains numerous scrambling and de-scrambling methods. However, these methods are not designed to leave the header information alone during the scrambling and descrambling process, thus they are unable to retrieve information about the scrambled audio without de-scrambling all of the information.

The following detailed description addresses various of the just-cited issues, redressing some of these problems and providing new functionality not heretofore contemplated.

## Brief Description of the Drawings

FIG. 1 is a flowchart showing acts employed in an illustrative embedding technique.

FIG. 2 is a block diagram showing an apparatus used to embed or retrieve data using the method of FIG. 1.

FIG. 3 is a flowchart showing acts employed in an illustrative decoding technique.

FIG. 4 graphically displays operation of a first illustrative embodiment.

FIG. 5 is a flowchart showing the embedding of data in accordance with the first illustrative embodiment. The dashed lines show interaction with the auxiliary data.

FIG. 6 is a flowchart showing the decoding of data. The dashed lines show interaction with the auxiliary data.

FIG. 7 graphically displays operation of a second illustrative embodiment.

FIG. 8 is a flowchart showing the embedding of data in accordance with the second illustrative embodiment. The dashed lines show interaction with the auxiliary data.

FIG. 9 is a flowchart showing the decoding of data. The dashed lines show interaction with the auxiliary data.

FIG. 10 demonstrates aspects of an illustrative embodiment in conjunction with digital compression techniques.

FIG. 11 A and B are two block diagrams showing an embedding and retrieving apparatus in accordance with an illustrative embodiment.

FIG. 12 shows an illustrative embodiment of the apparatus of FIG. 2 for embedding data.

FIG. 13 is shows an illustrative embodiment of the apparatus of FIG. 2 for retrieving the data.

FIG. 14 shows a block diagram for an enabling process referenced in the discussion concerning attack resistance.

FIG. 15 shows the block diagram for a registration process.

FIG. 16 demonstrates the way in which dynamic locking blocks the duplication of the auxiliary data.

FIG. 17 displays the input and output for an exclusive-or (XOR) function.

FIG. 18A displays an overview of a process of dynamic locking and embedding of the auxiliary data.

FIG. 18B displays an overview of a process of retrieving and dynamic unlocking of the auxiliary data.

FIG. 19A shows the modification step of dynamic locking for locally masked embedded data.

FIG. 19B shows the modification step of dynamic locking for pulse width modified (PWM) embedded data.

FIG. 19C shows the modification step of dynamic locking for embodiments based upon PN sequences. (Auxiliary data is abbreviated as aux.)

FIG. 20A displays the pseudocode in the form of a flowchart for locking and embedding the auxiliary data using header blocks.

FIG. 20B displays the pseudocode in the form of a flowchart for retrieving and unlocking the auxiliary data using header blocks.

FIG. 21 shows the basic process behind the example utilizations. (The dotted boxes are optional. The dashed boxes group similar items. In addition, although three key locations are shown, usually only one key is used and its location depends upon the utilization requirements. Finally, the

abbreviation ID is used and many times refers to an identifier, but can also refer to any auxiliary information.)

FIG. 22 shows an apparatus that may be used for these robust data embedding techniques.

FIG. 23A shows an embodiment of the apparatus of FIG. 22 for dynamic locking.

FIG. 24B is a block diagram showing an embodiment of the apparatus of FIG. 22 for dynamic unlocking.

FIG. 25 is an overview of a process of automatic ID management.

FIG. 26 is the pseudo-code for implementing an exemplary automatic ID management process.

FIG. 27 is an apparatus to implement automatic ID management.

FIG. 28 is a portable MP3 audio player containing the apparatus of Fig. 27.

FIG. 29 is an overview of a process employing two watermarks.

FIG. 30 displays the pseudocode for the embedding process of Fig. 29.

FIG. 31 displays the pseudocode for the retrieving process for Fig. 29.

FIG. 32 displays an apparatus that may be used in connection with the process of Fig. 29.

FIG. 33a is an overview of a scrambling process, where dotted boxes are optional.

FIG. 33b is an overview of a descrambling process, where dotted boxes are optional.

FIG. 34a is the pseudo-code for an exemplary scrambling or descrambling process.

FIG. 34b shows the input and output for the exclusive-or (XOR) function.

FIG. 35 shows an exemplary apparatus for performing the scrambling or de-scrambling processes.

FIG. 36 is an overview of a degradation and recovery process.

FIG. 37 is the pseudocode for the degradation and recovery process of FIG. 36.

FIG. 38 is a simple and efficient example of the degradation and recovery process using a threshold crossing and adjusting only the next point.

FIG. 39 is the pseudocode for the degradation and recovery process of FIG. 38.

FIG. 40 is an overview of an apparatus suitable for implementing the process of FIGS. 36-39

## Detailed Description

### *Introduction to Data Hiding Arrangements*

The following discussion proceeds with reference to exemplary embodiments, methods, and operational features. Except as explicitly stated, or reasonably indicated, the examples given should not be taken as precluding other arrangements.

In accordance with one embodiment, a method and apparatus of data hiding and retrieval is provided that offers high efficiency, with an attendant reduction in cost. In some embodiments, psychophysic data hiding is used – without the need to modify or transform the original data – in order to identify the locations at which the data should be hidden. In certain embodiments the encoding leads to essentially no detectable change in the content statistics, making the hidden signal still harder to identify and remove.

The technology can be implemented so as to permit the user to set parameters that vary the perceptibility, robustness, and embedding rate, permitting the disclosed technology to be used in a broad variety of applications.

An exemplary apparatus includes a logic processor and storage unit, such as those that come with a standard personal computer or on DSP boards. These devices act as data readers, comparer and data writers, so that the user's desired watermark can be embedded and/or retrieved.

An exeplary process involves embedding and retrieving auxiliary information into original data to produce combined data. One or more detection criteria can be used to determine where in the original data to locate and/or adjust data points so as to carry the auxiliary information. The detection criteria can be used to locate positions - referred to as local masking opportunities - in the original data at which the embedding of auxiliary data will produce less perception, as compared to other simplistic processes.

When embedding the auxiliary data, the data points in the original data are investigated in accordance with the detection criteria to determine the existence of local masking opportunities. The detection criterion or criteria may involve, for example, comparing the data point to a predetermined value and examining the relationship of the data point to nearby points. If the detection criteria are met, one or more of the nearby points, or the data point being investigated, is changed to indicate the value of an embedded bit of auxiliary data.

Thus, although the search for local masking opportunities typically progresses point by point through the data, the investigation of each point may include not only the value of that point, but also values of one or more nearby points and/or one or more relationships among the points. If the investigation of a point shows the existence of a local masking opportunity, data is embedded by setting the value of one or more of the local points, i.e., either the point being investigated or one or more of the nearby points.

The value to which the nearby data points are set in the illustrative embodiment is typically dependent upon the data point being investigated, as well as on the value of the auxiliary data bit. The data point value can be set so that it has a specified relationship with the neighboring data points. The process is continued until the original data has been traversed or no additional auxiliary data remains to be embedded.

Retrieving the auxiliary data is the inverse of the embedding process. The combined data is traversed using the detection criteria to locate the local masking opportunities. As each local masking opportunity is located, the nearby data point or points that was/were set to indicate the embedded bit is/are read to extract the embedded data. The process is continued until the combined data has been traversed.

In the preferred embodiments, a data point or points are set to a value relative to the nearby data points, and not to an absolute value. Both setting data points at the local masking opportunity and setting the data point to a value related to the nearby points, rather than to a value unrelated to the original data, provide masking that reduces the perceptibility of the data. The data is extracted by determining the relationships or values of the point or points near the local masking opportunity.

For the two preferred embodiments described in detail below, only points with large values are adjusted, and by a minimal amount; thus, these embodiments are based upon the masking of a weak stimulus by an intense stimulus. The process is applicable to analog and digital data. However, both embodiments are explained in terms of digital media due to current switch to digital media and the ease of understanding.

Specifically, the first preferred embodiment uses the difference between a data point after a peak and the peak level to carry auxiliary information, as long as the peak is above a large threshold and the original difference between the peak and next point is not too great. This large threshold and minimal differences produce the desired perceptual masking. The embedding process adjusts the point after the above-threshold peaks to hide the auxiliary data. Correspondingly, the retrieving process measures the difference between each above threshold peak level and the next data point to retrieve the auxiliary data.

The second preferred embodiment uses the change in slope across a positive, large, steep, threshold crossing to hide the auxiliary information, as long as the original change in slope is not too great yet steep enough to accept the ensuing adjustment. Again, the large threshold produces the desired perceptual masking. In the implementation, the embedding process adjusts the change in slope to embed the data, whereas the retrieving process measures the change in slope to obtain the auxiliary data.

Usually, the preferred embedding process spectrally implicitly spreads the energy of the auxiliary information throughout the original data. This broadband approach produces data that is more difficult to remove than sub-band approaches that place the data in an inaudible frequency range. If desired, parameters can be chosen so that the process produces protected data that is statistically identical to unmarked data. Importantly, the process can be adjusted to produce the desired tradeoffs between perception, coding rate and robustness to attack.

Such embodiments preferably – although not essentially - operate on the original data without requiring any complex data transformations, such as a Fourier transformation. Thus, if the original data represents information in the time domain, the data can remain in the time domain as the auxiliary data is embedded and retrieved. Of course, the technology can operate on original data of all types, such as in the frequency or time-frequency domain. For example, it can be applied to MPEG data, including the MPEG 1 and 2 specification, ISO 11172-3 and ISO 13818-7 respectively, which exists in the time-frequency domain.

Finally, the problem of bit-rate reducing techniques, known as compression, removing the watermark can be bypassed by using separate, but possibly identical, watermark procedures during the compression (a.k.a. encoding) and decompression (a.k.a. decoding) process.

*Exemplary Embodiments and Methods*

A system according to one embodiment comprises a method and apparatus for hiding auxiliary information (or data) in original data and for retrieving the auxiliary information.

FIG. 1 is an overview of the steps involved in carrying out an illustrative method to embed data. FIG. 2 shows a block diagram of an apparatus **10** that may be used to perform the method of FIG. 1.

Apparatus **10** includes a logic processor **14**, which can be a general purpose microprocessor, such as an Intel Pentium or DEC Alpha, of the type a personal computer or engineering workstation, a digital signal processor (DSP), such the the Texas Instruments TMS320 line, a specialized CPU, such as a media processor, or a custom processing circuit. Apparatus **10** also includes a storage unit **18,** which can include random access memory (RAM) or delays. Because the algorithms used in the illustrative embodiment are not computationally intense, they require calculations on the order of less than one million instructions per second and can be performed by most modern personal computers, and one many less capable devices as well (e.g., personal digital assistants, dedicated media players, etc.).

The original data mentioned below may represent sound that is recorded by sampling its amplitude periodically, with each sample using binary numbers to represent the magnitude of the sound at a particular time. Likewise, the samples may represent pixels of an image or video. Still further, the original data can be any series of binary data associated into groups. Similarly, the illustrative auxiliary information is any data that can be represented as "1"s and "0"s, but other symbol alphabets can likewise be used with corresponding adaptation of the disclosed arrangements.

FIG. 1 shows that in step **20**, a portion of the original data is read into storage unit **18** of FIG. 2. Step **24** shows that the sample data is investigated sequentially by the logic processor **14** to locate sample points that meet predefined detection criteria. Such sample points indicate the existence of "local masking opportunities," because the detection criteria are such that a change in the value of the sample or a few samples at or near that point to embed auxiliary data will usually have minimal perceivable by the listener of the sound. The amount of masking will depend upon the data type and settings chosen by the user. For example, the masking will be great for uncompressed audio and less for bit-rate reduced (digitally compressed) audio such as MPEG. The same detection criteria will be applied during data retrieval to locate the hidden data.

Each point in the original data is preferably investigated to determine whether it represents a local masking opportunity. The criterion or criteria for determining local masking opportunities may entail not only the value of the point being investigated, but may also include the value of at least one nearby or neighboring point, or the relationship between the nearby point and the point being investigated. The detection criteria can require, for example, that the point being investigated exceeds a certain threshold value and/or that the point be a local maximum or peak, and/or that the point is a point of local maximum in a first- or higher-order derivative. The criteria may include a requirement that a point subsequent to the point being investigated have a value that differs from the point being investigated by less that a prescribed amount, or have some other relationship to the point being investigated.

The sample data points can be considered as plotted on a graph, for example with time on the x-axis and the magnitude of the sample on the y-axis. Thus, the series of data points can be considered as having a slope between any points, and the value of the slope can be part of the detection criteria. The criteria may specify, for example, that a slope defined by the point being investigated and a preceding point exceed a particular value, or that the change in slope before and after the point not exceeds a

particular value. The criteria could include any combination of requirements; the detailed examples are not essential or restrictive of the scope of the detailed technology.

In the illustrated cases, no complex data transformation is required to mask the auxiliary data, so comparing a point to the detection criteria is relatively quick and inexpensive. Unlike many prior art methods, which use distant points to convert the original data into the frequency domain to determine how to mask embedded data, the illustrative embodiment can determine masking opportunities using only nearby or neighboring points, i.e., points that are too close to use to determine useful frequency data. Nearby points include points that are next to the point being investigated or within a relatively small number of points, preferably less than 50 and more preferably less than 20. The criterion can be as simple as determining whether the point exceeds a threshold.

Step **26** shows that when a point meeting the detection criteria is located, the value of a specified sample point or sample points near the local masking opportunity is changed to reflect the value of the auxiliary information to be embedded. Although the changed sample may be simply set to a particular value to signify the value of the embedded bit, the new value typically depends upon the value of both the auxiliary data and the neighboring point or points that were investigated to detect the local masking opportunity. For example, the point may be set so that the change in value or slope signifies whether the embedded bit is a "1" or a "0" (or other symbol).

When a point is set to its new value, it is preferable that either the change does not prevent the original sample point from continuing to meet the detection criteria, or that this local masking opportunity is skipped and not detected in the retrieval process. Otherwise, the embedded auxiliary data may not be retrievable.

Alternatively, it is possible to merely embed the auxiliary bit as the least significant bit, or other, preferably low order, bit. The embedded bit is still masked because the location of the embedded bit was chosen to represent a local masking opportunity, such as when the data is larger than a prescribed threshold.

Step **30** shows that the process is ended at step **32** if no additional auxiliary data needs to be embedded. Otherwise, step **34** shows that if there is additional data in memory, the search for local masking opportunities continues. Step **36** shows that if all data in memory has not yet been searched, additional data is read into memory. Skilled persons will recognize that some overlap of the data in memory may be required to prevent missing local masking opportunities that occur at the beginning or end points of the data in memory.

FIG. 3 broadly shows the steps involved in carrying out a decoding method. Because the same processor and memory that were used to embed the data can be used to retrieve the data, although not necessary, the steps of FIG. 3 will describe extracting data using the hardware components of FIG. 2. Step **50** shows that a portion of the original data is read into storage unit **18**. Step **52** shows that logic processor **14** investigates each data point to determine the existence of a local masking opportunity. If a sample point meets the local masking opportunity criteria, step **54** shows that the embedded "1" or "0" bit of auxiliary data is extracted using the inverse relationship of how the auxiliary data was embedded. Step

**56** shows that if additional combined data is in the memory, the logic processor continues to investigate the remaining points with step **52**. Step **58** shows that if all the data in memory has been investigated, but there is uninvestigated combined data in the data file, additional data is read into memory in step **50**. Step **60** shows that the process is ended when all the combined data has been investigated.

Two particular embodiments are described briefly here, and in detail below, to demonstrate the flavor of this methodology. As shown in FIG. 4, the first embodiment uses large, positive peaks as the detection criteria **120** and the auxiliary information is stored in the difference **130** between the peak and the next point. As shown in FIG. 7, the second embodiment uses large, steep threshold crossings with minimal change in slope as the detection criteria **140**, and the auxiliary information **150** is carried in the change in slope.

The methodology is applicable to analog or digital data, even though the preferred embodiments use digital data. For example, analog data can be sampled at the Nyquist rate to produce digital data in which additional information is hidden. Then, the combined digital data can be returned to the analog domain by any existing method known in digital signal processing (DSP). The analog data now contains the embedded data, which can be decoded by using sampling. This is just one possible method to encode analog data with the above methodology.

The methodology is also applicable to audio, speech, images, video or any other perceivable signal. With audio and speech, the original data may represent pressure versus time, magnitude versus frequency, or a specific frequency magnitude versus time. With images, the original data may represent gray code versus space, separate or combined RGB or equivalent values versus space, or magnitude versus frequency. Video data encompasses the image data with an added dimension of time available. For example, with MPEG bit-reduced audio or images the auxiliary data may be embedded in scaling factors or frequency coefficients versus frequency or time or both.

Usually one of the detection criteria is a large threshold. With 16 bit audio, a threshold greater than 48 dB above the minimum value is desirable. This threshold allows the data to be changed with minimal perception due to masking. Masking is the psychological term defined as the increase in threshold for steady-state stimuli. Use of the term in this disclosure is much broader than that definition, and describes how one set of data reduces the perception of other data. Specifically, for uncompressed, magnitude-time data, the sensitivity of the sensory system decreases with increased input level, thus the small adjustment of a neighboring data point is masked by the large value of the threshold. For bit-rate reduced, time-frequency data, such as MPEG data, the masking is minimal and more similar to the textbook definition since masking has been used to reduce the bit rate.

Finally, this method is applicable to data where masking is not used, but the efficiency of the process in that it is does not require a key, such as a PN sequence, or original data for retrieval of the auxiliary information, is an advantage. In summary, the parameters of the detection criteria will determine the interaction between the data rate, process complexity and perceptual quality.

*Embodiment 1*

The first particular embodiment is based upon hiding the auxiliary information in large peaks within the original data. In this embodiment, the auxiliary information is preferably broken into N bit words, with synchronization data placed between the words for better error recovery. The auxiliary information does not need to include sync pulses between the words if robustness to noise or modified files is not needed.

FIG. 4 conceptually shows that the first embodiment detects a peak or local maximum and sets the value of the subsequent point in relation to the peak to indicate the value of the embedded bit.

FIG. 5 includes the pseudocode in the form of a flowchart for the embedding process. The process begins by searching the original data until a positive peak that lies above a large threshold, labeled thr, and has a relatively small decrease after the peak, labeled dS, is found. This process is demonstrated in boxes **200**, **210** and **220**. The detection criteria are checked in the most computationally efficient order, which includes first checking to see if the point represents a peak since peaks are the least likely criterion.

When a desirable peak is found, the data point after the peak is adjusted according to a user defined bit depth, b, to carry the auxiliary information. Specifically, if it is the beginning of an auxiliary word, the synchronization code is embedded by adjusting the point after the peak, x[n+1], to be equal to the peak, x[n], minus half of the maximum allowable change, dS/2, between the peak and the next point, as shown in boxes **242**, **230** and **250**. An auxiliary information bit of one is encoded by adjusting the point after the peak, x[n+1], to be equal to the peak, x[n], minus half the maximum change, dS/2, and plus the half the bit depth magnitude, $2^{b-1}$. Correspondingly, an auxiliary information bit of zero is encoded by adjusting the point after the peak, x[n+1], to be equal to the peak, x[n], minus the sum of half the maximum change, dS/2, and half the bit depth magnitude, $2^{b-1}$. This embedding of zeros and ones is shown in boxes **242**, **240**, **260**, **270** and **280**. The next two points after embedding the data should be skipped so one does not create another peak for very slow changing (i.e. flat) data, as shown in box **290**.

These steps are repeated until the auxiliary information, box **242** and **240**, has been hidden in the original data or the original data is finished.

FIG. 6 displays the pseudocode in the form of a flowchart for the retrieval process of the first particular embodiment. The process begins by searching the original data until a positive peak that lies above a large threshold, labeled thr, and has a relatively small decrease after the peak, labeled dS, is found. This process is demonstrated in boxes **300**, **310** and **320**. Again, the search first looks for a peak to improve efficiency.

When a desirable peak is found, the difference between the peak and the data point after the peak is measured to retrieve the auxiliary information. Specifically, if the peak minus the point after the peak, x[n]-x[n+1], is close to half of the maximum allowable change, dS/2, a new auxiliary word is beginning, as shown in boxes **330** and **350**. If the peak minus the point after the peak, x[n]-x[n+1], is approximately equal to half the maximum change, dS/2, minus half the bit depth magnitude, $2^{b-1}$, an auxiliary bit of one is found. If this difference, x[n]-x[n+1], is close to the sum of half the maximum

change, dS/2, and half the bit depth magnitude, $2^{b-1}$, an auxiliary bit of zero is retrieved. This retrieving of zeros and ones is shown in boxes **340**, **360**, **370**, **380**, and **382**. The two points immediately after retrieving the data can be skipped as shown in box **390**.

These steps are repeated until the auxiliary information has been retrieved in the original data or the original data is finished.

There are three user-defined parameters, including threshold, thr; bit depth, b; and maximum allowable change after the slope, dS. For 16 bit audio, the threshold is usually around 48 dB above the minimal quantization, as discussed above. For data with more bits per sample, the threshold may be increased to reduce perception. The bit depth is an indication of the relative change to be made to the sample point to embed the data. Thus, the smaller the bit depth, the less disturbance of the original data, making the embedded data less perceptible to the listener, but less robust, that is, more susceptible to being lost to noise or attack. Minimal perception in 16 bit audio is found when bit depths are between 1 and 6 bits. However, higher bit depths can be used if one desires more robustness to noise in trade for more perceptual degradation. The maximum allowable change after the peak, dS, must be at least the desired bit depth magnitude, $2^b$. On the one hand, one can gain better robustness to noise at the expense of more distortion, if dS is set to twice the bit depth magnitude, $2^{b+1}$. On the other hand, if one desires to keep the threshold undetectable to statistical cryptoanalysis (labeled statistically invisible), dS should be set at $2^b$, and b should be small, probably below 3 bits. If dS is not $2^b$, one can use the discrepancy of the average difference between large positive peaks and their next points between embedded file and regular file data to determine if the file contains embedded data or not. Finally, if dS is much greater than $2^b$, the auxiliary information embedding rate will be increased, because more peaks will be found suitable for data embedding. Using the principles explained above, skilled persons will be able to set the user-defined parameters to values appropriate to the requirements of a particular application.

As discussed above, the large threshold usually reduces the perceivable effect of adding the auxiliary information, and may even cause the auxiliary data to be non-perceivable, depending upon the data type. In addition, many data points satisfy the small difference between the peak and data point after the peak, because with a slope near 0 at the peak, the data is changing the least. This small difference means that the adjustment will be small as compared to the threshold, thus reducing the chance of perceiving the embedded auxiliary data

The pseudocode is shown using a buffer with what appears to be look ahead capabilities (i.e. x[n+1]). This makes the process easier to explain and understand. However, the process is causal, as determined by replacing n+1 with k, and keeping track of the last two points, x[k-1] and x[k-2].

Finally, one can add more criteria to define the peak. For example, the peak extends for one more point each direction where x[n]>x[n-2], x[n]>x[n+2], x[n]>x[n-3], x[n]>x[n+3], and so on, or the peak is of minimal sharpness, i.e. x[n]-x[n-1]>5. Both of these criteria produce better robustness to noise and less distortion since it will take more noise to move the location of the peak, although changes in the peak criteria affect the rate at which auxiliary data can be embedded.

The embedded data density and bit rate will vary with the original data and with the user-defined parameters. For example, bit rates of between 99 and 268 bits per second were achieved in CD quality audio data using a bit depth of 5 and a threshold of 5,000 (74 dB). Using a bit depth of 8 and maintaining a threshold at 5,000, the average embedding rate was 1,000 bits per second. When the threshold is lowed to 2,000 at a bit depth of 8, an average embedding rate of 2,000 bits per second was found.

*Embodiment 2*

The second particular embodiment hides the auxiliary information in large, steep threshold crossings which do not have a large change in slope. The method is more robust to noise changing the detected location. This occurs because it is less likely that noise changes the location of a threshold crossing as compared to a peak, since a threshold crossing usually has a slope larger than the slope at the peak, which, by definition, has a slope near zero. Testing with audio data has shown this embodiment, as compared to the first embodiment, to produce a lower embedded data rate and is more perceivable at a lower bit depth, in trade for the robustness to noise. One will probably find the optimal embodiment dependent upon the application.

FIG. 7 conceptually shows that data is embedded by setting the slope after the threshold crossing in relation to the slope at the threshold crossing.

In FIG. 8, the pseudocode for hiding the auxiliary information using the second preferred embodiment is presented in the form of a flow chart. The process begins by searching the original data until a positive, large, steep threshold (labeled thr) crossing with minimal change in slope (labeled dS) is found. This process is demonstrated in boxes **400**, **410** and **420**.

When the desirable threshold crossing is found, the data point after the threshold crossing is adjusted according to a user defined bit depth (b) to carry the auxiliary information in the change in slope. Note that the change in slope is defined as (x[n+1]-x[n])-(x[n]-x[n-1]), or equivalently as x[n+1]-2*x[n]+x[n-1]. Specifically, if it is the beginning of an auxiliary word, the synchronization code is embedded by adjusting the point after the threshold crossing, x[n+1], so that the change in slope is zero, as shown in boxes **442**, **430** and **450**. An auxiliary bit of one is encoded by adjusting the point after the threshold crossing, x[n+1], so that the change in slope is positive by an amount equal to half the bit depth magnitude, $2^{b-1}$. Correspondingly, an auxiliary bit of zero is encoded by adjusting the point after the threshold crossing so that the change in slope is negative by an amount equal to half the bit depth magnitude, $2^{b-1}$. This embedding of zeros and ones is shown in boxes **442**, **440**, **460**, **470** and **480**. The point after embedding the data can be skipped for efficiency as shown in box **490**.

These steps are repeated until the auxiliary information has been hidden in the original data or the original data is finished.

FIG. 9 shows the pseudocode in the form of a flowchart for the retrieval of the auxiliary information in the second preferred embodiment. The process begins by searching the original data until a positive, large, steep threshold (labeled thr) crossing with minimal change in slope (labeled dS), is found. This process is demonstrated in boxes **500**, **510** and **520**.

When a desirable threshold crossing is found, the change in slope around the threshold is measured to retrieve the auxiliary information. Again, the change in slope is defined as (x[n+1]-x[n])-(x[n]-x[n-1]), or equivalently as x[n+1]-2*x[n]+x[n-1]. Specifically, if the threshold crossing has almost zero change in slope, a new auxiliary word is begun, as shown in boxes **530** and **550**. If the threshold crossing has a positive change in slope approximately equal to half the bit depth magnitude, $2^{b-1}$, an auxiliary bit of one is found. If the threshold crossing has a negative change in slope approximately equal to half the bit depth magnitude, $2^{b-1}$, an auxiliary bit of zero is retrieved. This retrieving of zeros and ones is shown in boxes **540**, **560**, **570**, **580**, and **582**. The point after retrieving the data can be skipped for efficiency as shown in box **590**.

These steps are repeated until the auxiliary information has been retrieved in the original data or the original data is finished.

As mentioned above, one does not want the embedding process to eliminate the embedded location from fulfilling the detection criteria. Specifically, in this embodiment, the pre-threshold change condition, $x[n]-x[n-1]>dS+2^{b-1}$, in the detection criteria of box **420** and **520** requires that the adjustment of the next data point does not bring the point back below the threshold. An alternative approach, is to ignore this condition and to set either the current or next point (x[n] or x[n+1], respectively) to the threshold if the embedding process would cause the next point to move below the threshold, and ignore any data points that are equal to the threshold in both the embedding and retrieving process. Interestingly, only when embedding a sync or 0 could the next point move below the threshold. Given these options, the described embodiment is chosen so the process is causal, thus incorporating the known advantages of causal processes.

Once again, the large threshold and maximum allowable change in slope condition, dS, reduce the perception of embedding the auxiliary data, and depending upon the data type can cause the embedding process to be completely non-perceivable. The maximum allowable change in slope condition, dS, can have any value. A larger value allows a higher data rate with more perceivable distortion, whereas a smaller value produces minimal distortion with a lower data rate. Our preferred setting for dS in 16 bit audio is equal to the bit depth magnitude, $2^b$. Again, bit depths below 6 bits produce minimal distortion, but higher bit depths can be used for robustness to noise and attack.

Using a threshold of 2,000 (i.e. 66 dB) and a bit depth of 5, data rates between 40-100 bits per second are expected, with an average of about 75 bits per second, for CD quality audio. At a bit depth of 8, the bit rate increases to an average of 100 bits per second.

*Modifications*

Particular embodiments have been described in detail above. However, there are many simple modifications that can be made to optimize the process for each use.

In some applications, a very simple embodiment could use a simple threshold to determine a local masking opportunity and then encode the auxiliary data in the LSB of the point exceeding the threshold or of another point in the vicinity of the point exceeding the threshold. Such a variation is

extremely simple, yet provides reduced perceptibility compared to prior art LSB schemes. As with the other embodiments, one must ensure that changing the value does not remove the point for the detection criterion. In this case, one could simply skip embedding where the change brings the data below the threshold, and change the current value of the data point to the threshold so that the data point will be skipped in the retrieving phase.

To increase the robustness to attack or noise, the following changes could be made. (Attack is defined as a person or machine trying to remove the auxiliary information from the combined signal without distorting the perception of the original data.)

Using a dynamic threshold can make it harder to remove the auxiliary information. An example dynamic threshold is an offset sinusoidal waveform. When using a dynamic threshold, dS should be small and close to $2^b$ so that the process does not change the distribution of the differences between neighboring points, i.e. be statistically invisible; thus, an attacker cannot use this data to find the threshold.

One can also use the statistical gaps when dS is larger than $2^b$ to find the threshold if the attack uses a DC shift. A DC shift is obviously a more potent attack for the second preferred embodiment than the first, but could affect the first preferred embodiment since threshold is one of the detection criteria.

(Attack-resistant methods are further considered below.)

The process may use more global definitions for peaks and threshold crossings, for better robustness to noise. Specifically, a peak or threshold crossing definition may be used that includes more points on each side.

Finally, the process can use any type of error correction in the auxiliary information to increase the robustness.

To increase the data rate, the following changes may be made. The auxiliary information does not need to include the extra sync pulses between the N-bit words, especially if robustness to noise is not needed. In addition, negative going peaks and/or more thresholds can be used to increase bit rate. Finally, the process can use more than a binary system in adjusting the second bit to encode more information. However, the result is more likely to be perceivable or less robust to attack.

An interesting twist is to embed different auxiliary information on positive and negative peaks, and/or on various thresholds. In addition, with stereo files, the channels can be coded separately, or encoding can move between channels with consecutive points moving between left and right channels.

A change that can improve the perception is to move the data point after the embedded point towards the value of the embedded point if combining the auxiliary information causes a large value change in the embedded point.

As mentioned above, the data does not have to be relative to time. For example, the data may represent magnitude versus frequency. In addition, the data could be viewed as magnitude of a specific frequency versus time. All frequencies may be included for an increased data rate. In other words, embedding may be performed in the spectrum or spectrogram. Advantageously, one doesn't have to change the format of the given data to use this process and corresponding apparatus.

Consider, for example, bit-reduced data, such as MPEG compressed data. MPEG-compressed data comprises a series of data points that represent scaling factors and frequency coefficients. Auxiliary data may be embedded in the series of MPEG data points, using, for example, one of the two particular embodiments described above. When using the first particular embodiment, one may want to increase the peak or modify its LSB such that the term is only increased, rather than decrease the point after the peak, such that quantization error is not increased in the MPEG data, especially when dealing with scaling factors. Skilled persons will recognize that, in using data like MPEG data that is divided into time frames, one may use, for example, scaling factors or frequency coefficients from consecutive frames, as well as data points representing scaling factors and coefficients of different frequencies within a frame, when determining where to embed data. For example, the coefficients for a particular frequency in consecutive frames could be considered as a series of consecutive data points, and those data points analyzed in accordance one of the embodiments above to determine where to embed data in the series. In an alternative example, the series of data points representing scaling factors or frequency coefficients for different frequencies within a frame may be analyzed in accordance with the first or second particular embodiments above to decide where to embed data.

*Example Utilizations*

Below are included some example utilizations of an illustrative algorithm to aid in its understanding. This list is not complete, and only highlights the usefulness of the disclosed technology. Moreover, the applications given below are not reliant on the particular form of encoding used – any other form of digital watermarking, steganography, or data hiding, can alternatively be used.

The process can be used to embed copyright information. This information may include a code to determine if the data can be copied. Copying devices, such as CD writers, can include an inexpensive integrated circuit that could interpret embedded data and prohibit copying.

In addition, author's or artist's name and affiliation can be embedded. In this utilization, the auxiliary information is small and would be repeated over and over with synchronization pulses between each duplication. Alternatively, the copy code could be embedded using embodiment 1, and the creator's name and affiliation using embodiment 2 (i.e., several embedded data may co-exist in a work).

Such technology can be used to send additional information. This information may be transmitted in ASCII or ANSI with 8 bit "words" (not to be included with digital words being defined as 32 bits) and synchronization pulses between these words, if desired. The information may be a secret message, lyrics to the song, or a description of the artwork. For lyrics, this could be useful for karaoke machines and CD or DVD players.

*Digital Compression*

The main problem with hiding data and digital compression (reducing bit rate not dynamic range) is that the process of hiding data is at odds with digital bit-rate reducing techniques known as compression (a.k.a. encoding and decoding). This incompatibility occurs since the goal of data hiding is

to make the data minimally perceivable and the goal of compression is to remove minimally perceivable parts.

To this end, FIGS. 10A and B demonstrates an illustrative process for data hiding, if at some point the data must be compressed. For example, this may happen while transmitting the data.

In FIG. 10A, the auxiliary information is embedded in the non-compressed data using the above-detailed process or any other method, as shown in box **600**. Then, when the data needs to be compressed, the auxiliary information is retrieved via the above-detailed or the other appropriate method, and re-embedded in the compressed data with the above-detailed process or any other method, as shown in box **610**. The algorithm for data hiding in the compressed and non-compressed data may be the same algorithm, differing by only using different original data. Or they may be different.

In FIG. 10B, the auxiliary information is retrieved from the compressed data by the above-detailed or other method, the data is uncompressed, and the auxiliary information is embedded in the uncompressed data, as shown in box **620**. Finally, when needed, the auxiliary information can be retrieved from the data using the above-detailed or other method, as shown in **630**. Once again, the algorithm for data hiding in the compressed and non-compressed data may be the same algorithm, differing by only using different original data, or not.

*Apparatus*

As described above, FIG 2 demonstrates that the detailed process can be implemented via logic processor and storage unit **18**. FIG. 12 shows the implementation with a digital processor **1200** and digital memory **1210**. The digital processor **1200** may be defined as the equivalent of a digital signal processor (DSP), general-purpose central processing unit (CPU), or a specialized CPU, including media processors. A likely DSP chip is one of the Texas Instruments TMS320 product line. A CPU could include one of Intel's Pentium line or Motorola/IBM's PowerPC product line. The design is straightforward for someone familiar with the state of the art given the pseudocode in Figs 5 through 9.

In addition, as shown in FIG. 13, a person familiar with the state of the art could implement the process with analog and digital circuitry, either separate or in an application specific integrated circuit (ASIC). The analog and digital circuitry could include any combination of the following devices: a digital-to-analog converter (D/A), comparators, sample-and-hold circuits, delay elements, analog-to-digital converter (A/D), and programmable logic controllers (PLC). Programmable logic arrays (PLDs) can likewise be used. Someone familiar with the state of the art given the previous description and pseudocode in FIGs. 5 through 9 could easily design the circuit.

FIGs. 11A and B show that the logic processor and storage unit typically comprise an embedding apparatus **700** and retrieving apparatus **770**. The embedding apparatus **700** includes the following. A data reader **710** to read original data **720** and auxiliary data **730**. A comparer **740**, that is, a circuit or device for comparing data points with known values or other data points. A data writer **750** to write the combined data **760** to a permanent or temporary storage media.

The retrieving apparatus **770** includes the following. A data reader **715** to read the combined data. The data reader **715** may be identical to the embedding data reader **710**, but it also may be different. A comparer **745**, that is, a circuit or device for comparing data points with known values or other data points and, if necessary, producing the auxiliary bit or bits. Once again the comparer **745** may be identical or different that the embedding comparer **740**. A data writer is not always necessary since the auxiliary information may be taken from memory or only displayed for the corresponding use.

*Ramifications and Scope*

As the reader can see from the description above, and can demonstrate by testing the process with CD quality audio, the above-detailed processes and apparatuses for hiding auxiliary information within original data are efficient and have configurations which are non-perceivable. These advantages are mainly due to finding locations to hide the auxiliary data without needing to transform the signal to the frequency domain, so that masking may block or reduce the perception of the auxiliary data.

*Corruption- and Attack-Resistance*

As noted, a further aspect of the technology detailed herein relates to methods for increasing the resistance of embedded data to corruption and attack. For expository convenience, the word "attack" is used herein, but is meant to include both deliberate efforts to remove embedded data, and incidental removal of such data. Attack may include duplication, which is defined as being able to replicate or impersonate the embedded data from one data segment to another. Attack may also include modification, which is defined as changing the embedded data for a desired affect, such as from "no copying" to "copying allowed".

The below-detailed technology sets forth two embodiments describing ways of using embedded data such that the embedded data is more robust to attack: the enabling and registration process. In addition, embodiments improving the robustness of embedded data to duplication or modification are disclosed, including dynamic locking and unlocking.

The first embodiment utilizes an enabling process, which involves using embedded data to enable an action, such as copying, playing or otherwise rendering. Thus, if the embedded data is removed by attack, the end-user has gained nothing because the original data has become unusable. Improvements in this process occur when the embedded data is robust against duplication and modification.

The second embodiment utilizes a registration process, where the recording device embeds its registration in the data. In this embodiment, the recording device can refer to a physical device, such as a CD or DVD burner, or virtual device, such as an MP3 or AAC encoder. This registration process allows any illegal media to be traced back to the original owner assuming that recording devices are registered when purchased. At the very least, the illegal media may be traced back to the specific recording device's place of purchase, providing law enforcement with a good starting point.

The dynamic locking and unlocking embodiments improve the robustness of existing or future embedded data techniques to duplication and/or modification. Dynamic locking causes the embedded data to be dependent upon the media, e.g., by including one or both of the following steps. The first step includes modifying the auxiliary information by the media. The second step includes encrypting the auxiliary information, possibly modified in the first step. The encryption technique may be RSA, DES or any appropriate algorithm. After dynamically locking the auxiliary information, it is embedded in the original data. Each step of dynamic locking provides its own independent advantages. However, incorporating both steps produces auxiliary information that cannot be transferred between media, modified, or created.

The dynamic unlocking process performs the inverse steps, assuming each specific step was performed in the dynamic locking process. The first step involves decrypting the retrieved data. The second step involves unmodifying the output of the first step or the retrieved data directly, depending upon whether the first step was performed, and thus producing the original auxiliary data.

Five exemplary utilizations of the enabling and dynamic locking process and apparatus are described briefly here and in detail below to aid in the understanding of both processes and apparatus. These utilizations include (1) distribution of compressed media such that it can only be played by the requester's playback device, (2) using the presence of the embedded data to specify copy-once access, (3) protection of DVD media, (4) photo-card validation, and (5) sending secure secret messages.

In the first example utilization, a media player, such as a computer with MP3 software player, contacts an Internet site to download media, such as a song in MP3 format. The player sends its unique identifier to the Internet site, where the identifier is modified using the original data and the result is encrypted. The modified and encrypted identifier is then embedded in the original data, and the combined data is downloaded to the player. The media player is able to extract the identifier from the combined data, and compare it to its own identifier. If these identifiers are identical and any additional information, such as a date limit, is verified, the player will play the data. If the combined data is copied to a second player having a different identifier, the second player will not play the combined data.

If an unauthorized person were able to determine the identifier, he could then embed it in other songs and play them on his player. By encrypting the identifier, an unauthorized person would be unable to determine the identifier, even if he were able to extract the auxiliary data from the combined data. In addition, if the process did not include modifying the auxiliary information with the original information, the embedded data could be copied between media. Finally, the encryption key also requires proper handling, and the identifier may include additional information besides the player identifier.

An example of the second utilization includes, rather than a unique identifier, a predefined copy code such as "allow no copying," "allow copying one time, but not copying of a copy," and "allow unlimited copying." The recorder would retrieve the copy code and not copy unless permitted by the code. The copy would either contain no "allow copying one time..." code, or contain an "allow no copying" code. For broadcasts, both the player and the broadcast unit would know the code beforehand (i.e. predefined) or the code would be included in the broadcast.

In the third example utilization, two approaches are described. In the first approach, a DVD player will not play the DVD without retrieving the predefined identifier embedded in the original data. For extra security the identifier could be encrypted with a key located at a central database or in a section of the DVD not available for copy. In the second approach, the identifier could control the number of generation of copies allowed, noting that if no identifier exists, no copies can be made. Or, there could be two layered identifiers for both types of copy management.

The fourth example utilization involves embedding secure data in the picture of a photo-card, as in a photo used for identification purposes like a driver's license or credit card. If the retrieved information at the photo-card reader does not match that of the central database, the card is recognized as a fake and will not be authorized for use. Note that the information and key exchange must be securely transmitted.

The fifth example utilization allows the secure transmission of secret information, hidden in the media. Most bystanders will not know the secret message is attached. If found, the hidden information cannot be read by, modified by, and/or transferred to other media by an imposter when the embedded data is dependent upon the media and encrypted. Different types of encryption, symmetric or public/private key, can be used for creating the desired protection or authentication of the embedded data. This hidden information enables a person or machine on the receiving side to perform an action.

The exemplary apparatus for these processes involves a logic processor, possibly including DSP chips, host CPUs or custom analog or digital circuitry, and memory. The configuration and machine code are easily designed given this disclosure and familiarity with the state of the art in cryptology and electrical engineering.

Before turning to a detailed exposition of the foregoing, consider some definitions. Media or content includes, but is not limited to, audio, video, still images, combinations of the above, and forms related to other senses. The terms media and content are used interchangeably. Media does not refer to a storage medium. A media or content segment includes, but is not limited to, a song, part of a song, movie, part of a movie, part or all of a sound track, part or all of a still image, a taste, a touch, and an odor. Original data is the raw, unprotected data. The auxiliary information refers to any data that is to be embedded in the original data. The ID **140** in Fig. 21 refers to this auxiliary information, and may include but is not limited to, information such as the player ID, number of copies allowed, usage time or date limits, and content enhancement information such as author, copyright, publisher, song lyrics or image details. The embedded data is the data that is actually embedded in the original data. The embedded data differs from the auxiliary data by the transformation used in the embedding process. This transformation can include the modifying process and/or encryption involved in dynamic locking, and the embedding process such as bit manipulations, pulse-width modulation, or spreading with frequency transformation or pseudo-random noise sequence. The combined data results from adding the embedded data to the original data. Robustness to attack is defined as getting around what the embedded data is supposed to provide or prevent. Finally, a pirate is a person who tries to illegally obtain the data or use the protected device.

*Enabling Process*

Fig. 14 demonstrates an exemplary enabling process. This process uses a logic processor **900** and memory **910**, as shown in Fig 22. First, as shown in box **10**, processor **900** retrieves the auxiliary data from the combined data **5** and stores it in memory **910**. Then, processor **900** determines whether the embedded data allows the desired action, as shown in box **20**. If so, the desired action is allowed, as shown in box **30**. If not, the desired action is disallowed, as shown in box **40**.

*Registration Process*

Fig. 15 demonstrates the registration process. This process involves assigning a unique registration code **305** to each recording device **300**, and embedding the registration code **305** into the media when it is recorded, as shown in box **310**. Then, when illegal media is found in an open-market **320**, it can be traced to the owner of the recording device via the registration code **305**, as shown in box **330**.

The process is similar to gun registration assuming the recording device is registered upon purchase. At the very least, the illegal media could be traced back to the recording device and its place of purchase, thus aiding law enforcement.

This recording device may be a physical device or virtual device. A physical device could include a CD or DVD burner. A virtual device could include a software program using processor **900** and memory **910** to digitally compress (bit rate reduce) audio, such as a MP3 ripper or AAC encoder. Remember that media refers to the perceived data and not the storage medium.

*Dynamic Locking*

Fig. 16 displays the way in which dynamic locking blocks the duplication of the auxiliary data. Duplication is blocked for both bit-for-bit copying of the embedded data between content, and retrieving the embedded information, and re-embedding it into different content, such that the different content appears authentic.

Specifically, when only modifying the auxiliary information, a pirate will not be able to move the embedded data from one media segment to another without figuring out how to correctly unmodify and re-modify the embedded data. When only encrypting the auxiliary information, a pirate will not be able to obtain the auxiliary information. The pirate will be able to retrieve the embedded data, but not decipher it since it is encrypted. When both steps, the auxiliary information cannot be moved from one media segment to another. If it is moved directly, the modification step of dynamic locking causes the embedded data in the new media to be incorrectly unmodified because the values in the new media segment used for unmodifying the retrieved data don't match the values in the original media where the data was modified. If the pirate tries to unmodify and re-modify the embedded data (since the details of this step may be known), he/she must first have the key to decrypt the data in order to move it to new media segment.

Fig. 17 displays the input and output for the exclusive-or function (XOR). The XOR is its own inverse and extremely efficient.

Fig. 18A displays an overview of the dynamic locking and embedding process. The whole process contains three steps, but either one (not both) of the first two steps, i.e. those steps of dynamic locking, can be skipped. However, when both dynamic locking steps are performed the difficulty in duplicating the data is improved. In addition, the order of the last two steps can be switched. This switch is beneficial when the content, including the embedded data, is encrypted, usually for other content protection reasons, or when the modification step has some of the desirable features such as requiring a key to be unmodified.

In the first step, box **600**, the auxiliary data (d), which is to be embedded, is modified based upon the original content (c). This step is designed to modify the auxiliary data to be dependent upon the original content such that the embedded data cannot be copied bit-for-bit between content. The chosen content bits should be critical to the content, such that they cannot be changed in new content to make it appear authentic. A desirable function is the exclusive-or (XOR) operator since this function is its own inverse and efficiently implemented on digital processors.

In the second step, box **610**, the modified data is encrypted such that the original auxiliary bits cannot be obtained from the embedded data. Thus, the original auxiliary bits cannot be re-embedded in different content, making this different content appear authentic. If the auxiliary data is not modified by the original content before being encrypted, it could be copied bit-for-bit from the original content to new content making the new content appear authentic. Any existing or future methods of encryption, including DES and RSA, can be used, with known methods of key management, all of which is well described in the prior-art.

In the third step, box **620**, the encrypted and modified (labeled dynamically locked) auxiliary data is embedded into the original content.

Fig. 18B displays an overview of the process used to retrieve and dynamically unlock the auxiliary data. The whole process contains three steps, and each step should only be performed if the corresponding step was performed when the data was embedded. In addition, if the order of the last two steps was switched while embedding, these two corresponding steps should be switched during this retrieval process.

In the first step, box **630**, the embedded data is retrieved from the content. At this time, the embedded data consists of encrypted and modified auxiliary data (assuming both dynamic locking steps were performed). In the second step, box **640**, the retrieved data is decrypted. In the third step, box **650**, the output of step two is unmodified. The result is the original auxiliary data.

In addition, dynamic locking and unlocking can use correlated data. Correlated data may include information such as song lyrics or the address of the person in a photographic identification card.

Fig. 19 shows several example implementations of the modification part of dynamic locking and unlocking when data is embedded such that it will not be perceived (i.e. watermarking). Although, only the modification part is shown in Fig 19, the modified auxiliary information may be encrypted before

being embedded and decrypted after being retrieved (but before being unmodified), if desired. In addition, the modification of the auxiliary information may be skipped, and the auxiliary information may be only encrypted before being embedded and decrypted after being retrieved. The cryptology process is not discussed in detail since someone familiar with the state of the art easily understands its implementation.

Fig 19A shows dynamic locking and unlocking as applied to an apparatus earlier-described. For dynamic locking, the peak value, box **200**, or threshold crossing value, is used in the exclusive-or (XOR) calculation to modify the next N auxiliary information bits, where N is the number of bits per sample in the data (such as 16 bits for CD audio). Then, these modified N bits of the auxiliary information are optionally encrypted and embedded (e.g., by the above-detailed methods) using locally masked bit manipulations of difference $\Delta$. This process is repeated for the next group of N peaks, and so on, until the whole modified auxiliary information is embedded or all the original data has been used with modified auxiliary information being repetitively embedded.

The embedded data can be retrieved using the process described above, decrypted (if required), and unmodified. The unmodifying process is the inverse of the modifying process. Since the XOR function is its own inverse, the peak values of the combined data and the decrypted auxiliary information are applied to the XOR function. Importantly, the peak values are identical to those of the original data since they were not changed during the embedding process.

For example, when using CD-audio, N is 16 bits. Thus, for this example, the first 16 bits of the auxiliary information are modified by the first peak value using the XOR. Then, these modified auxiliary information bits are optionally encrypted and embedded in the data points after the current peak and the next 15 peaks. This process is repeated for the following group of 16 peaks and auxiliary information bits, and so on, until all the data is embedded or all the original data has been used. The modified and optionally encrypted auxiliary information can be embedded over an over again within the data, by restarting the process with the first 16 bits of the auxiliary information after all of the bits have been embedded.

The embedded data can be retrieved, decrypted (if encrypted), and unmodified with the inverse of the XOR calculation, which is an XOR calculation. Thus, the first original 16 bits of the auxiliary information can be obtained by performing the XOR calculation with the retrieved and decrypted embedded data and first peak value. The retrieving process is continued for the next group of 16 peaks of the combined and embedded data, and so on, until the whole auxiliary information is found or all of the combined data has been traversed.

It is very important to keep proper track of the position of the groups of 16 bits in the auxiliary information when modifying for embedding, and unmodifying after retrieving.

The above-detailed technology allows sync pulses in the combined data. These sync pulses can be used to align the auxiliary information with the value used in modifying the auxiliary information. For example, rather than embedding data after the peak used to modify the auxiliary information, a sync pulse could be embedded and used for re-alignment during the retrieval process.

Fig 19B shows dynamic locking and unlocking as applied to Patent #5,774,452 "Apparatus and method for encoding and decoding information in audio signals" by Jack Wolosewicz of Aris Technologies, incorporated herein by reference. For this case, the data values occurring previously in time to the embedding of the pulse-width modulated (PWM) bit stream and shown in box **220**, could be used in an XOR operation with auxiliary information to modify and unmodify the embedded data. In this case, several data values would need to be used to modify all of the auxiliary information. For example, when using 16 bit data and embedding 256 bits of auxiliary information, the dynamic locking and unlocking process would use the previous 16 original data points to modify all of the auxiliary information. As long as the data is received in the same order as it was embedded, it does not matter if the data values used to modify the auxiliary information overlap with the previous embedded bit stream. If one finds a configuration where the above does matter, it can easily be handled by skipping the second embedded bit stream and marking it as skipped in the combined data.

Fig 19C shows an overview of applying dynamic locking and unlocking to embedded data schemes based upon pseudo-random noise (PN) sequences. In one embodiment, the PN sequence could skip the $M^{th}$ data point, as shown in boxes **250** and **270**, where M is equal to the number of bits per sample in the data (N) times the length in bits of PN sequence segment applied to each auxiliary information bit. This $M^{th}$ data point would be used in an XOR operation with b bits of the auxiliary information to modify the auxiliary information. For example, lets assume each auxiliary information bit is embedded with a 1024 bit segment of the PN sequence in 16 bit audio and the auxiliary information is 64 bits long. Then, after adding the PN sequence to 16384 (M = 1024 bit PN segment*16 bit audio) bits of original data, another original data point is skipped to modify the auxiliary information. It will take 4 (64 bit auxiliary information/16 bit audio) of these segments to embed each auxiliary information. Equivalently, 4 adjacent original data points could be skipped every 65536 (1024 bit PN segment*16 bit audio*4 PN segments) original data points and embed the whole modified auxiliary information in one continuous stream of four PN segments.

This modified and optionally encrypted auxiliary information can be used to control the fashion in which the PN sequence is added to the original data, as well known in the state of the art of spread spectrum technology. Specifically, in many applications, the PN sequence will be phase shifted by the modified auxiliary information (i.e. where 0 scales and adds the negative value of the PN sequence and 1 scales and adds the positive value) or simply multiplied by the auxiliary information. Once retrieved, the modified auxiliary information could be unmodified using the inverse XOR calculation with the skipped data point.

Another embodiment for PN sequences is using the skipped data point to modify the next N bits of the PN sequence, not the auxiliary information. If one point is skipped, the number of PN bits modified, M, should be equal to N, the number of bits in the data. If two points are skipped, M is equal to 2*N, and so on. Modifying the PN sequence using an XOR calculation and optional encryption is one scheme. However, this may reduce the randomness of the PN sequence, and other modification functions

can be employed to maintain randomness. Finally, the modified and optionally encrypted PN sequence is embedded in the media data and used to retrieve the embedded data.

In a final implementation of dynamic locking, it is applied to embedding methods that use PN sequences to determine where to place the auxiliary information in the original data, possibly after being transformed into the frequency domain. Such methods include that of patents #5,613,004 and #5,687,236 "Steganographic method and device" by Marc Cooperman and Scott Moskowitz of the Dice Company, incorporated herein by reference, and patent-pending technology of AT&T labs (Lacy J, Quackenbush SR, Reibman AR, Shur D, Snyder JH. (1998) "On combining watermarking with perceptual coding." ICASSP'98 Seattle, WA.), detailed in EP889471, incorporated herein by reference. For these methods, the PN sequence used to embed the data could be required to never have more than N continuous embed bits and start with a non-embed bit, where N is again the number of bits per sample in the original data. Then, the original data point adjacent and previous to the first embed bit modifies the next N bits of the auxiliary information. This process may be repeated until all the original data is embedded, such that the modified auxiliary information is embedded repetitively. For example, when embedding in the frequency domain from low to high frequency with 16 bit data and a 32 bit auxiliary information, the non-embed bit in the frequency bin just below first embed bit is used to modify the next 16 auxiliary information bits. Then, the non-embed bit in the frequency bin just below the $17^{th}$ embed bit is used to modify the next 16 auxiliary information bits. Next, the non-embed bit in the frequency bin just below the $33^{rd}$ embed bit is used to modify the first 16 auxiliary information bits, and so on.

In a similar scheme, the PN sequence could be applied to every other or $k^{th}$ (where $k < N$ bits per sample in original data) data point, such that no limitations need be applied to the PN sequence. The process guarantees to have a non-embed bit next to the $N^{th}$ embed bit, and is implemented is similar fashion the previous method.

For all these methods of dynamic locking using PN sequences, the dynamic unlocking process is the inverse and obvious to a person familiar with the art given the previous disclosure.

Fig. 20 demonstrates applying dynamic locking and unlocking to data embedded in header, not content, data. Fig. 20A displays the pseudo-code for the dynamic locking process. In general, the auxiliary data bits, of length L, are locked and placed in the header of frames of the content and repetitively embedded.

Specifically, the process of Fig. 20A starts at the beginning of the content bits (box **700**) and auxiliary data bits (box **705**). Then, L auxiliary data bits are locked by being modified with L bits of the content using the XOR or applicable function, and/or encrypted (box **735**). These L content bits should be critical to the either or both file format and content, such that they cannot be replicated in a different media segment without disturbing it. Next, M bits of locked auxiliary data are embedded in the frame header (box **710**). These M bits should be less than L, and preferably L is divisible by M, such that the L bits are embedded in L/M frame headers. If L is not divisible by M, a person familiar with the state of the art can easily handle the offset. Then, the content is checked to see if more frames exist (box **715**). If there are no content frames left, the process is completed (box **730**). If there are more content frames, the

auxiliary data is checked to see if any previously modified bits exist (box **720**). If there are previously modified auxiliary bits left, the next frame is read (box **725**), and the process is continued at box **710**. If there are no previously modified auxiliary bits left, the next content frame is read (box **740**), the auxiliary data is re-started at bit 0 (box **705**), and the process is continued at box **710**.

This process assumes the auxiliary information is of length L and L is reasonably short for ease of explanation. It is obvious that if you have a very large number of auxiliary bits, you can break them into segments of length L, and rather than starting at the first auxiliary bit each time, start at the $k^{th}$ segment. To this end, the auxiliary bits are embedded within the data, broken into segments of length L and each segment is embedded in L/M frame headers.

To increase robustness to attack, a pseudo-random noise (PN) bit sequence could be used and the first N critical content bits with a corresponding PN bit value of 1 could be used for modifying the auxiliary information.

Alternatively, only the first important M content bits in each frame, rather than L bits every L/M frames, are used in the XOR calculation when embedding M locked auxiliary data bits in each frame. In this case, the auxiliary data bits are modified in each frame, specifically, between boxes **725** and **710** in Fig 20A. Once again, a PN sequence could be used to randomize which M bits of original audio are used. Importantly, M must be large enough so that error correction in new content cannot repair all the content bits that need to be changed, such that a bit-for-bit transfer of the auxiliary data makes the new content appear authentic. The value of M depends upon the frame size and desired bit rate.

When using compressed content, such as MPEG data, specifically Layer III (MP3) or AAC audio as specified in the MPEG2 specifications, including the MPEG 1 and 2 specifications, ISO 11172-3 and ISO 13818-7 respectively, herein by reference, the frames and header bits are pre-defined. The private, copyright, or ancillary bits can thus be used to embed the data. When using content without pre-defined frames, such as in raw PCM audio, databases, or software applications, the frames can simply be created. For example, the content could be arbitrarily divided into 1024 bit frames with header bits for the embedded data.

Alternatively, the locked auxiliary data could be placed only in the global header, defined as the header for the complete file, or in a linked but separate file. These two cases are less secure than embedding the data throughout the file. More bits mean the data will be more robust to attack via brute force. For broadcast content, the data should be embedded throughout the content as described above so the rendering device or person can receive the auxiliary information and respond accordingly from any point in the broadcast.

Fig. 20B displays the pseudo-code for the retrieval and dynamic unlocking process for the auxiliary data embedded in Fig 20A. In general, the auxiliary data bits are retrieved by reading them from the header of the content frames and unlocking them, in a repetitive manner.

Specifically, the process of Fig. 20B starts at the beginning of the content bits (box **750**) and auxiliary data bits (box **755**). Then, N content bits are saved, such as in memory **910** of Fig. 22, so they can be used to unlock the next N retrieved auxiliary data (box **785**). Next, M bits of locked auxiliary data

are read from the frame header (box **760**). Then, the content is checked for existing frames (box **765**). If there are no content frames left, the process is completed (box **780**). If there are content frames left, the auxiliary data bits are checked to see any exist (box **770**). If there are auxiliary bits left, the next frame is read (box **775**), and the process is continued at box **760**. If there are no auxiliary bits left, the retrieved auxiliary data is unlocked (box **790**), the next frame is read (box **795**), the auxiliary data is re-started at bit 0 (box **755**), another N content bits are saved (box **785**) and the process is continued at box **760**.

For this example, unlocking the retrieved embedded data (box **790**) involves performing an XOR operation (since it is its own inverse) on the N content bits that were saved in box **785** and the last N retrieved embedded data bits, and decrypting, if required. In addition, since the data is repetitively embedded in each frame, the retrieving process must overlay the auxiliary data bits after the last bit as received (box **790**) and make sure the auxiliary data bits do not change throughout the file. If the auxiliary data bits change throughout the file, the file is not authentic.

Alternatively, if another modification function was used, its inverse should be used. Importantly, the same retrieved auxiliary bits and original content bits should be used in the inverse calculation as were used in the modifying calculation. For the embedding example where the first M audio bits of the frame were used to modify the auxiliary data, the first M audio bits of the frame should be used to unmodify the modified auxiliary data, which was retrieved and decrypted. If a PN sequence was used to modify the auxiliary data, the same PN sequence should be used to unmodify the data.

If an alternative embedding step was used, the auxiliary bits are retrieved accordingly. For example, if bits are embedded in the global header or linked file, the are read from the global header or linked file, respectively.

Finally, the appropriate steps should be taken if the auxiliary data is longer than L or L is not divisible by M. These steps are obvious to a person familiar with the state of the art given the above explanations about dynamic locking and unlocking.

*Example Utilization*

These five example utilizations are described to aid in understanding the enabling and dynamic locking process and apparatus. The general underlying process for these examples is displayed in Fig. 21 and the corresponding apparatus is shown in Figs. 22 and 23. The process, in general, begins with a sending device **100,** dynamically locking an ID **140** as shown in box **110**, and embedding the locked ID within the media as shown in box **120**. Remember, as defined at the beginning of this section, the term ID usually refers to an identifier, but can include any auxiliary information. The sending device **100** may be an encoder, recorder, transmitter, storage medium, or the like.

The media is then transmitted to a receiving device **130** in which the locked ID is retrieved as shown in box **160**, and dynamically unlocked as shown in box **170**. Then, the proper action is enabled if allowed by the retrieved **ID 140**, as shown in box **180**. The receiving device **130** may be a decoder, player, recorder, and/or the like.

When the dynamic locking and unlocking processes include encryption and decryption, the encryption key must be located somewhere and transmitted safely, as shown in boxes **151**, **152**, and **153**. Transmitting the key safely is well understood by one familiar with the state of the art in cryptology. The location of the key depends upon the requirements of the utilization. The five utilizations demonstrate various key locations. For most utilizations, the key will be available only in one of the three possible locations. In addition, the encryption and decryption key will usually be identical (symmetric), and referred to as the encryption key in the discussion below. However, public/private key encryption could also be used in many of these situations. When discussing private/public encryption below the key will be specified as the public or private encryption key. Finally, certain utilizations may not need to transmit the auxiliary information since the values are predefined.

In addition, the use and location of **ID 140**, the types of sending devices **100** and receiving devices **130** are also explained in more detail in these example utilizations.

The five example utilizations include distribution of MP3 data, copy-once access to broadcast data, DVD copy protection, photo-card verification, and secret data transmission. From these explanation, many more utilizations are obvious.

Regarding distribution of MP3 data, the concept is explained using several scenarios. All scenarios include both a software PC-based and portable MP3 audio player, and distribution via the Internet.

In the first scenario, the MP3 data exists on the Internet and is purchased by an end-user. The delivery system interacts with the end-user's player, securely transmitting ID **140** and the encryption key, shown in box **151**, from the receiving device to the sending device, and dynamically locking, including encryption, the ID **140** in the MP3 data. In this scenario, the encryption key, shown in box **151**, is located on the end-user's player. After the MP3 file is delivered (i.e. downloaded) only the end-user's player can play the data since other players will have different IDs. A portable and PC-based player may share the ID **140**, and this is easily implemented by a software program and current digital electronics, such as EPROM or flash memory. Since the ID **140** is dynamically locked the end-user cannot extract the ID **140** and use it in another song or MP3 file.

In another scenario, the MP3 encoder and player may be part of one software program, which transforms CD, DVD or broadcast audio into MP3 audio with the embedded data containing the dynamically locked, including encryption, ID **140**. In such an example, there is no need for the exchange of an encryption key, as displayed in box **151**, and ID **140**. The software applications should be programmed such that the key and ID **140** are protected from the end-user, as well known in the state of the art in software. Again, the key, shown in box **151**, is located within the end-user's player. The transformed MP3 audio is now only playable on the end-user's system and/or portable player, and it is not possible to move the ID **140** to another song as described above.

In yet another scenario, the key could be located in a central database, as shown in box **152**. This configuration allows a different key for each player and MP3 audio sample. This configuration

increases robustness to attack since new keys are used for each song, but involves extra management tools and responsibilities.

In a final scenario for MP3 audio, the ID **140** could contain time limits for listening to the audio or a date limit that, when exceeded, the audio will not play. The player will keep track of how many times the song has been played or whether the date has expired. The ID **140** could contain a demo code, which does not limit the song to one player.

Regarding copy-once access (defined as allowing an end-user to copy the media only once, perhaps for time shifting purposes), the concept is explained in terms of the broadcast of a movie. With broadcast media, it is best if everyone shares the same encryption key. The key, as shown in box **153**, could be broadcast embedded in the movie, and changed for each broadcast. In addition, if the embedded data is not encrypted, there is no need for a key, thus simplifying transmission. Finally, the copy-once ID **140** will be predefined, meaning that it is already defined in the transmitting and receiving device, as shown in Fig. 21 where ID **140** has an optional location in the transmitting device. Once the broadcast is received and the retrieved ID **140** enables the data to be recorded, the recorder can record the movie and either remove the copy-once ID **140** or change the ID **140** to a predefined code that informs other recorders that the media has been copied once.

Regarding DVD copy protection, there are two scenarios. In the first scenario, the player will not play the media unless the embedded ID enables the action. The encryption key, as shown in box **153**, is be included on the DVD in a non-copy access location. This means the user will be able to play the media only when the DVD disk is present since the player will not play the DVD data without retrieving the correct ID. A copy of the entire DVD (minus the encryption key since it is unable to be copied) or a copy of a content file will be unusable since the key to decrypt the embedded data will not be found and without it the player will not work.

In addition, the key could be located in a centrally accessible database, and possibly linked to the requesting end-user player, as shown in box **152**. This configuration increases robustness to attack since access to the key is monitored, but includes extra management responsibilities for the content provider and additional time for the end-user. The key could also be purchased and encrypted by the key in your player as described in U.S. patent 5,933,498 to Paul Schneck (incorporated herein by reference). Once again, the ID **140** will be predefined, and exist in the sending device **100**.

In a different scenario, the predefined ID **140** could be used to enable the recorder, and allow a certain number of copy generations, or a copy of only the original, known as serial copy management. ID **140** could be modified to allow one less recording generation each time the DVD is recorded. Possibly through keeping track of the recorded generation and originally allowed count or by reducing the allowed count. For serial copy management, the watermark could be removed in the second generation DVD. Remember that in this approach if the watermark does not exist, no copies can be made. Finally, there could be two-layered ID **140**s for both types of copy management.

The photo-card utilization example involves having the picture in the photo-card embedded with the ID **140**. If the correct information is not present, the card is a fake and will not be authorized for use.

To increase the security of the method dynamic locking is applied; the ID **140** is reversibly modified by the photograph or connected data such as the corresponding name and address, and encrypted, such that the information cannot be copied between cards or from a legitimate card to an illegal card. The matching ID **140** and encryption key can be stored at a database only accessible by every sending device (i.e. in the sending device) and securely transmitted between the database and the photo-card reading device, such as using RSA key exchange or any other method known in the state of the art of cryptology. Besides being as secure as other cryptology techniques, another advantage of this process is that it requires transmission of minimal data, including the short ID **140** and encryption key.

The last example utilization allows the secure transmission of secret information in ID **140**, hidden in the media. Most bystanders will not know the secret message is attached. Once the receiving device extracts the hidden message, the receiving device, a connected device, or a human will be enabled by the hidden information contained in ID **140**. If found, the hidden information can be protected from being moved to other media segments and/or interpreted by using dynamic locking with various encryption schemes. For example, if the secret information is encrypted with your public key, only you can recover it. Or if it is encrypted with your private key, people or devices receiving the message using your public key know it was signed by you and is authentic. If it is encrypted with a symmetric key, only the holders of the key could have created and read the message. Finally, if the modification step of dynamic locking is used, the receiver knows the message was not transferred from a different media segment.

*Apparatus*

Fig. 22 shows an exemplary apparatus used to implement the enabling, registration, and dynamic locking processes. The hardware includes a logic processor **900** and memory **910**. The logic processor **900** may be defined as the equivalent of a digital signal processor (DSP), general-purpose central processing unit (CPU), or a specialized ASIC chip. A likely DSP chip is one of the Texas Instruments TMS320 product line. A CPU could include one of Intel's Pentium line or Motorola/IBM's PowerPC product line. The design is straightforward for someone familiar with the state of the art given the description of these processes. The memory **910** includes any type of memory.

Fig 23A shows more detail of the apparatus for dynamic locking. Specifically, the logic processor **900** and memory **910** must work together to act as the modifier **1010** and encrypter **1040**. Modifier **1010** performs the modification step of dynamic locking. Encrypter **1040** performs the encryption step of dynamic locking.

Fig 23B shows more detail of the apparatus for dynamic unlocking. Specifically, the logic processor **900** and memory **910** must work together to act as the decrypter **1045** and the unmodifier **1015**. The decrypter **1045** performs the decryption step of dynamic unlocking. The unmodifier **1015** performs the unmodifying step of dynamic unlocking. The unmodifier **1015** and decrypter **1045** of dynamic unlocking may use the same or different circuitry as the modifier **1010** and encrypter **1040** of dynamic

locking. However, when using the same circuitry, the dynamic locking and unlocking processes would use different control programs.

*Binding and ID Assignment*

As noted, another aspect of the technology detailed herein relates to media-binding, e.g., the fashion in which consumers legitimately access protected content while controlling piracy. A basic concept is that the content contains an ID that locks it to a particular user or broadcast and the rendering device automatically determines whether the content can be accessed based upon the current and previously rendered IDs and rules. Such technology may result in increased sales of content for the content providers.

One aspect of the technology resides in having the rendering device keeping track of the IDs contained in both the current and previously accessed content. This allows the rendering device to control access to new content based upon the new content's ID, the rules provided with the content (by the content providers) and/or within the device, and the IDs from previously rendered content by the device.

The ID may be linked to the user or the broadcast. User IDs work well for content that is sold for a user's continued use, whereas broadcast IDs work well for content recorded by the user from a broadcast.

An example implementation is as follows. For user-linked content, the rendering device includes constraints that limit the number of content tracks with different user IDs that can be accessed in a certain amount of time, possibly influenced by the number of times content with each user ID has already been accessed. For broadcast content, broadcast IDs and the optionally included rules can be used to limit rendering or copying of each broadcast. In other words, with broadcast IDs, the limits are based upon date or number of times that ID is played, not on the total number of broadcast IDs.

More specifically, a portable MP3 player can keep track of each song's user ID, and if the previously played songs contain more than N different user IDs, the player decides if it can replace an old user ID with the new one due to the old user ID's date and number of times songs with that ID have been played. Similarly, if a broadcast ID is contained in memory, the MP3 player notes that the user has played the audio X times and Y times is allowed by the broadcast, or the date is past the broadcast's allowable usage date.

To this end, it is easy for the consumer to use the device, as he/she is not required to posses an ID card. In addition, there is no need for a global database linking the user to the ID; thus, the user's privacy is not compromised. For example, if a user loses his/her ID, it can be obtained from previous content. However, the user or broadcast ID can be kept secret and other privacy methods can be used. Most importantly, access to the media is limited, as the content providers wish, but the user is not inconvenienced.

Again, a review of relevant terminology may be in order. The rendering device is a device that can play, view, record or perform a similar action upon the data. The rendering device can provide any

type of perceived data, including but not limited to images, audio and video. If the rendering device has a portable section, such as with a MP3 player, the loader, which puts the content onto the rendering device, is considered as part of the rendering device. The ID may be a user or broadcast ID. For example, many MP3 players can also record broadcasts, and these broadcasts will, in the future, contain embedded broadcast IDs, possibly as watermarks or header data with digital broadcasts. Content refers to the desired audio, video, image, or other relevant perceived data. Content providers include but are not limited to record labels, movie studios, and independent artists. The ID may be embedded within the content such as bits in the header file or a watermark, or the ID can be linked to the encryption and decryption of the content. Finally, this automatic ID management may be used in conjunction with other methods, such as media-binding.

Fig. 25 displays an overview of ane automatic ID management process. In the process, the rendering device **100** keeps track of the IDs contained within the content it has previously accessed (box **110**). The rules **120** may be provided in the device hardware and/or contained with the content. The rules **120** decide whether or not the device can access the new content based upon its ID (box **130**).

If the rendering device has a portable section, such as with a MP3 player, the loader, defined above as part of the rendering device, can be used to lower the amount of memory required within the portable section, thus lowering its costs. This means that with a portable rendering device, the portable section may contain all of the memory and processing hardware (described in detail below) required to perform this automatic ID handling, or the hardware may be split between the loader and portable section. For example, when a computer uses a software loader to put MP3 files onto a portable MP3 player, the loader may store all the information about IDs on the computer and all the rendering device needs to do is count the number of times each song is played and maintain date information for its current list of content.

Fig. 26 displays the pseudo-code to implement an example of the process. In this example, the rules **120** include constraints **245**, which are contained within the content as specified by the content provider, as well as default rules contained with the rendering device hardware. The constraints 245 are retrieved from the content **200** (box **240**). The constraints **245** may limit the number of content tracks with different IDs that a device can access during a set time-period. The constraints **245** may also change the time-period an ID is stored dependent upon the number of times content with a specific ID was accessed. The constraints **245** may be embedded within the content or attached as a header information or a linked file.

For ease-of-use, it is better to not change these constraints per song because it may confuse the user. Ideally, the constraints should be agreed upon and set in the rendering device. However, including the rules in the content is a viable option.

Before describing further details of this exemplary process, it is important to understand the illustrative apparatus for implementing the automatic ID management process (Fig. 27). The hardware includes a logic processor **300** and a memory **310**. The logic processor **300** may be defined as the equivalent of a digital signal processor (DSP), general-purpose central processing unit (CPU), or a

specialized CPU, including media processors. Again, a likely DSP chip is one of the Texas Instruments TMS320 product line. A CPU could include one of Intel's Pentium line or Motorola/IBM's PowerPC product line. The design of code for controlling logic processor **300** is simple for someone familiar with the state of the art given the above pseudo-code and description.

In addition, a person familiar with the state of the art could implement the logic processor **300** using analog and digital circuitry, either separate or in an application specific integrated circuit (ASIC). The analog and digital circuitry could include any combination of the following devices: digital-to-analog converters (D/A), comparators, sample-and-hold circuits, delay elements, analog-to-digital converters (A/D), and programmable logic controllers (PLC). Programmable logic arrays (PLDs) can likewise be used.

The memory **310** stores the information required by rules **120**, such as IDs, last play date, and the number of times that content with each ID has been accessed. Memory **310** may consist of standard computer random access memory (RAM). It is also desirable if memory **310** maintains this information even without power in the rendering device, perhaps but not limited to using ROM with backup and chargeable battery power, or memory that is stable without power, such as EPROM. As discussed above, memory **310** may consist of two separate modules when using a portable section and loader.

Now, back to a detailed description of the example process. It begins with the device **100** receiving new content **200**. From the content **200**, an ID **210** is retrieved. The ID **210** is checked to see if it is a user or broadcast ID (box **215**).

For user IDs, the following happens. If the ID **210** already exists in the memory **310** of device **100** (box **220**), the play count and last access date are updated (box **222**) and the content **200** is rendered (box **230**). If the ID **210** does not exist in memory **310** (box **220**), the rules **120** are checked. If another ID can exist in memory **310** (box **250**), ID **210** and the current date are added to the memory **310** (box **260**) and the content is rendered (box **230**). If another ID cannot be added, the rules **120** are checked to see if any existing IDs can be replaced because they are too old (box **270**). If any IDs can be replaced, the old ID is replaced with ID **210** (box **280**) and the content is rendered (box **230**). If no IDs can be replaced, the user may be warned and access to content **200** is denied or limited (box **290**). The user may also be presented with a link to buy the content (box **290**).

More specifically, the rules may allow a device to store 10 IDs, and IDs can be replaced if they have not been accessed for a week.

In addition, the number of times an ID has been rendered could be used to determine whether or not to replace the old ID with a new one (box **270**). This count value could influence the time period an ID is held is memory **310**; thus allowing ID **210** to replace a stored ID (boxes **270** and **280**). For example, if content associated with the stored ID has not been accessed in a week, it can be replaced. Conversely, if content associated with the stored ID has been played at least 7 times, it should be held for at least a month since its last access.

There are many other simple rules that can be designed to meet the specific needs of the content provider. Some may involve using difference equations to decide whether or not an ID can be replaced.

For example, the count for an ID can be reduced by one each day and incremented by one for each rendering of content containing the ID, and the ID can be replaced (box **270**) if the count is zero or less, or the date of last access is over a week.

For broadcast IDs, the following happens. The ID **210** is examined to see if it already exists in memory **310** (box **255**). If not, the ID **210** and current date are added to the rendering devices memory **310** (box **265**), and the content is rendered (box **230**). If the ID **210** does exist in memory, the play count, record date and/or last access date are checked to see if the content can be rendered (box **275**). The broadcast may allow only two renders, or one week of rendering, or rendering until a specific date. If the broadcast is allowed to be rendered, the count and last access date are updated (box **285**) and the content is accessed (box **230**). If the broadcast is not allowed to be rendered, the user is notified, the access is limited and a link to buy the broadcast or similar content may be provided, if applicable (box **295**).

In addition, the device should probably have some way to reset all of the information, such as IDs, date and count. The reset function may require a password code that is pseudo-random, thus requiring the user to contact support to reset the device. For example, the password may depend upon the day and year and obtained from an automation system. The reset button may also delete all the current content as well as ID information. This allows people to use one portable player with many friends at a party, but the loss of content will discourage piracy since it will be cumbersome.

Figure 28 shows a portable MP3 player **400** that contains the described apparatus implementing the described pseudo-code. In this case, the logic processor **300** could be a separate processor, or share access with the processor that decompresses the audio. The device also contains the necessary memory **310** to store the required information, such as ID, data and count, possibly even when the player **400** is without power. The device may share this memory with a software loader.

Finally, in any rendering device, the logic processor **300** could be a separate processor or share time with the processor handling content for the device, such as compressing or decompressing digital content.

*Multiple Watermarks*

Various advantages can accrue from using multiple watermarks instead of just one. In an exemplary system, one watermark is robust and declares that the media is protected. This watermark is embedded when the media is encoded into the desired format, such as MP3. This means that the intensity of adding the watermark is not an issue because the watermark is only added to the audio once, and copied with the audio by the distributor.

The other watermark declares that it is okay to play or record the media. It is efficient, and does not need to be difficult to remove, since removing it produces no advantageous results. The efficiency of this watermark is desirable since it must be embedded each time the audio reproduced, such as downloaded on the Internet, to link the media to the user, player, recorder and/or storage device. Thus, it greatly reduces the cost of copy management for the distributor. In addition, it lowers the cost of the players, since usually they only have to find this efficient watermark. Only when it does not exist, does

the player need to determine if the audio is protected with the robust but computationally intense watermark.

Most importantly, non-protected media may contain neither watermark and can be played by any device from any storage.

In more detail, Fig. 29 displays a process employing two watermarks. Media **100** exists in an insecure format, meaning that devices can play the media **100** even if it does not contain any copy protection and/or authentication watermarks. It is a format in which some artists wish to freely distribute their content, such as MP3. However, there are interested parties who don't want to distribute their media in the same format without allowing it to be freely copied and redistributed.

Watermark **110** declares that the media is protected. Watermark **110** must be extremely difficult to remove, and is allowed to be computationally intense. Many existing watermark methods meet this description, and future ones will certainly be designed.

Watermark **120** links the media to the user, player, recorder and/or storage device. This link determines if the user may copy and/or play the media. Watermark **120** must be a computationally efficient method that is hard to imitate.

Both watermarks are embedded at specific times in the reproduction process, as shown in Figs. 29 and 30. Watermark **110** is embedded when the audio is encoded, and copied with the audio when distributed. Thus, the computational intensity of adding the watermark is not that important. Watermark **120** is embedded when the media is reproduced, such as being distributed, placed on permanent storage, or encoded from an alternative form by a personal encoding device. The term reproduced refers to the legal transformation or distribution of the media, whereas copying refers to an individual producing an exact bit-for-bit replication of the media for legal or illegal utilization. Since watermark **120** is embedded every time the media is reproduced, its efficiency creates a reduction in cost. Since watermark **120** is embedded after watermark **110** it must be okay to layer the watermarks, as known to be possible with existing technology.

Optimally, the watermarks are search and retrieved in a specific order, as shown in Figs. 29 and 31. First, the media is searched for watermark **120** (box **300**). If watermark **120** is retrieved (box **310**) the embedded information is evaluated (box **320**). If the embedded information is correct, the desired action is enabled (box **330**). Alternatively, if the embedded information is not correct, the desired action is disabled (box **340**). Only if watermark **120** is not found does the media need to be searched for the computationally intense watermark **110** (box **350**). If watermark **110** declares the media protected, then the desired action is disabled (box **340**). If watermark **120** is not present (or declares the media to be free), the desire action is allowed (box **330**).

The just-detailed process can be used to restrict copying and/or playing of the media.

Fig. 32 shows hardware apparatus that may be used to implement the invented processes. The hardware includes a logic processor **400** and a storage unit **410**. The logic processor **400** may be defined as the equivalent of a digital signal processor (DSP), general-purpose central processing unit (CPU), or a specialized CPU, including media processors. A likely DSP chip is one of the Texas Instruments

TMS320 product line. A CPU could include one of Intel's Pentium line or Motorola/IBM's PowerPC product line. The design is simple for someone familiar with the state of the art given the above pseudocode and description. The storage unit **410** includes RAM when using a digital processor.

A person familiar with the state of the art could alternatively implement the process with analog and digital circuitry, either separate or in an application specific integrated circuit (ASIC). The analog and digital circuitry could include any combination of the following devices: a digital-to-analog converter (D/A), comparators, sample-and-hold circuits, delay elements, analog-to-digital converter (A/D), and programmable logic controllers (PLC). Programmable logic arrays (PLDs) can likewise be used.

*Content Scrambling*

As noted, it is often desirable to scramble content signals. The following discussion reviews certain improvements to such scrambling technology.

One such scrambling technique involves searching through the original digital data for detection criteria and then adjusting neighboring points to degrade the content, either without affecting the location of the detection criteria or affecting it in a known fashion so that the original signal may be recovered. The detection criteria may include the relationship between several points, or be as simple as a threshold crossing or include every $M^{th}$ point. The adjustment of the neighboring points may be as simple as multiplying the point after the threshold crossing by N. It is advantageous if N is less than one but not equal to zero so saturation and data points equal to zero are not a problem, and if the threshold is positive and the data is decreasing towards zero during the threshold crossing.

The process can include searching through the data for the detection criteria and then re-adjusting neighboring points to their original value. For example, if the adjustment in the degradation process uses multiplication by N, the recovery process multiplies by 1/N.

In the following discussion, digital content refers to digital data representing a perceived physical item, including but not limited to audio, video, and images. Digital data refers to the grouping of bits (1's or 0's) that represent a sample of the original digital content at an instant in time. Each bit group is equivalently referred to as a data point or sample. The data points are arranged in an order, many times representing a sequence versus time or frequency. In addition, the data points may be grouped again to form a subgroup, possibly used to represent a sequence versus frequency versus time, as is the case in MPEG standard compressed digital audio and video. Most importantly, the digital data has an order, with a beginning and end, such that searching the data is possible, and neighboring points can be defined as points close to each other. Finally, point(s) refer to one or several points.

Fig. 36 displays an overview of the degradation and recovery process, and Fig. 37 displays the corresponding pseudocode to be implemented by the apparatus.

To degrade the digital content (box **100**), the samples are searched for the detection criteria (boxes **200**, **210** and **220**). The searching stops after the last data point in the buffer has been examined (box **210**), and a new buffer may be presented if available. As known in the state of the art, data values must be saved between buffers and properly initialized for the first buffer so as the initial points are properly searched.

When the detection criteria are found, the neighboring data point(s) are adjusted so as to cause content degradation (box **230**). The adjustment of these points should not change the location of the detection criteria or change it in a known fashion; otherwise, the detection of the correct location to re-adjust the data to its original value (recovery) is not easy. In addition, it is desirable to prevent the adjustment from causing saturation or resulting in a value of zero, because then the original data point(s) will not be easily recoverable.

To recover the original digital content (box **110**), the degraded data is searched for the detection criteria defined by the degradation process (box **200**, **210**, and **220**). If the degradation process has changed the detection criteria in a known fashion, then the detection criteria in box **220** for recovery is different than that used in degradation. When the criteria location is found, the neighboring data point(s) are re-adjusted by the inverse of the method used in the degradation process (box **230**).

An example of this process is shown in Figs. 38 and 39. In this case (boxes **300** and **310**), the detection criterion is a threshold crossing (using c-notation: x[n-1]>thr && x[n]<thr) with a positive threshold (thr>0) while the data goes towards zero (boxes **400**, **410** and **420**). The neighboring point(s) include only the point after the threshold crossing (box **430**). To degrade the data, the adjustment involves multiplying the data point after the threshold crossing (x[n]) by N, where N is less than 1 (box **430**). By reducing the value of this data point, the detection criteria location is not changed. In addition, the closer N is to 0 (but not equal to 0), the more the digital content is degraded. To recover the original digital data, the point after the threshold crossing (x[n]) is multiplied by 1/N (box **430**).

There are additional simplistic detection criteria that can be used. For example, every $M^{th}$ data point may be degraded. In this case, synchronization for recovery may require scanning the data for M points until the correct degraded locations are found. In addition, peak values could be used, and the point after the peak could be reduced in value. As desired, this will not affect the detection criteria for the recovery process. Alternatively, threshold crossings with a negative threshold and the data moving towards zero are viable. Again, the data point after the threshold is reduced in absolute value towards but not equal to zero. For these last two cases, synchronization for recovery automatically occurs when searching the data.

Although, in the exemplary, the detection criteria do not change between degrading and recovering the original digital data, this is not a requirement. The detection criteria may change, if in a known fashion, such that the recovery process uses a different (but known) detection criteria than the degradation process. In other words, box **420** (or **220**, as discussed above) would be different for the degradation and recovery process.

The original content need not be represented by digital samples versus time, as one may have assumed. In many cases, such as using MPEG compression (i.e. MP3 audio), the digital samples represent subgroups of frequencies versus time. In this case the data may be searched across frequency for each subgroup, or across time for each frequency, or in any other but well defined combination. The data may also represent either the frequency magnitude or corresponding scaling factors.

Additionally, there are alternative ways to recover the data while removing most of the

perceptual degradation. For example, one could use a low-pass filter to recover the data. The recovered digital data will not exactly match the original digital data, but its perception may be acceptable. As well know by one familiar with the state of the art in DSP, filter characteristics such as type and order will affect the recovered data.

Alternatively, one could use pseudo-random sequences (a.k.a. a key) to set the detection criteria (box **220**) or the adjustment or re-adjustment of the data (box **230**). This randomness increases the difficulty to illegally recover the data. For example, a pseudo-random number greater than zero but less than one could be used as the scaling value N (box **430**). Or, a pseudo-random number between minimum and maximum threshold could be used for the threshold (box **420**). All that matters is that the degradation and recovery process use the same pseudo-random sequence. However, this configuration requires sending a key along with the data. The key may be embedded within the data using known techniques, such that the original data is still recoverable from the degraded data.

Fig. 40 shows illustrative hardware used to implement the described degradation and recovery processes. The hardware includes a logic processor **500** and a storage unit **510**. The logic processor **500** may be defined as the equivalent of a digital signal processor (DSP), general-purpose central processing unit (CPU), or a specialized CPU, including but not limited to media processors. A likely DSP chip is one of the Texas Instruments TMS320 product line. A CPU could include one of Intel's Pentium line or Motorola/IBM's PowerPC product line. The design of code for controlling logic processor **500** is simple for someone familiar with the state of the art given the above pseudo-code and description. The storage unit **510** includes RAM when using a digital processor, and is required to store the current buffer and/or previous point(s) for the detection criteria.

In addition, a person familiar with the state of the art could implement the logic processor **500** with analog and digital circuitry, either separately or in an application specific integrated circuit (ASIC). The analog and digital circuitry could include any combination of the following devices: digital-to-analog converters (D/A), comparators, sample-and-hold circuits, delay elements, analog-to-digital converters (A/D), and programmable logic controllers (PLC).

In accordance with another improvement to scrambling technology, a process is provided the avoids scrambling the header or other important information about the content. An advantage to leaving the header alone is that applications or devices can quickly read information about the content before de-scrambling and accessing the content. For example, with a scrambled MP3 file, a user can quickly learn about the song's length, artist, resolution, etc., before choosing to de-scramble and play it. Alternatively, the header may contain copyright information that the player is required to check before playing.

The scrambling process scrambles some or all of the non-header content. If only some of the non-header content is scrambled, it must be more than error correction, if present, can repair. When the content contains frames, each with its own header, as done with Motion Pictures Expert Group (MPEG) compressed audio or video, the header of each frame is avoided while scrambling some or all of the non-header content. The de-scrambling process recovers the original content from the scrambled content, similarly avoiding the header information.

An exemplary process involves using a pseudo-random noise (PN) sequence and the XOR function to scramble the content while avoiding the headers of each frame. The de-scrambler is identical since the inverse of the XOR function is the XOR function.

Again, a review of terminology may be in order. The header of a file contains important information about the file. This information may include the type of file, author, place of origin, date of origin, last modified date, file size, structure allocations, copyright codes, unique IDs, usage rules, etc. The header may exist only at the beginning of the file, at the beginning of frames within the file, or both. Frames are common for compressed digital media, such as MPEG audio and video. More specifically, with MP3 data, the header may include what the MPEG standard labels as header, error correction and side information. In addition, if the content does not contain frames or a header, such data can easily be created in a new structured file format.

Fig. 33a displays an overview of the scrambling process. If the file has only a global header or frames of known size without synchronization (sync) codes, the headers are located and skipped (box **105**) during the scrambling step (box **110**). In other words, there is no reason to look for the sync code (box **100**). The scrambling step may scramble part or all of the non-header content. If the file is broken into frames with additional sync codes, the sync codes that define the frames are found (box **100**), the header information is skipped (box **105**) and the content is scrambled (box **110**). Usually, the header contains information about the frame's size, which aids in locating the next sync code, as the sync code may also randomly occur within the data. Once again, the scrambling step may scramble part or all of the non-header content.

The scrambling step can consist of methods used in the prior art. Standard modern encryption, such as DES or RSA, is an excellent choice. With this encryption, although one may be able to de-scramble one file by brute-force, another file can remain secure even when using the same key. Other scrambling options may include simple mathematical operations with a PN sequence, such as multiplication, addition, subtraction, or exclusive-or (XOR). Division should be used carefully since it may cause bit error due to the imprecise nature of limited bit-length division.

Fig. 33b displays an overview of the de-scrambling process. De-scrambling is the inverse of scrambling, and only the content bits that were scrambled should be de-scrambled. If the file has only a global header or frames of known size without synchronization (sync) codes, the headers are located and skipped (box **155**) during the de-scrambling process (box **160**). In other words, there is no reason to look for the sync code (box **150**). If the file is broken into frames with sync codes, the sync codes that define the frames are found (box **150**), the header information is skipped (box **155**) and part or all of the remaining content in that frame is de-scrambled (box **160**). Usually, the header contains information about the frame's size, which aids in locating the sync code, as the sync code may not be unique and, thus, occur within the data. The de-scrambling step may de-scramble part or all of the non-header content, depending upon what was scrambled.

The de-scrambler should use the inverse of the function used by the scrambler. When scrambled with standard modern encryption, the de-scrambler requires a decryption key, and the key may be

different than the encryption key. For the mathematical operations, subtraction and addition are inverses, XOR is its own inverse, and division is multiplication's inverse. Division is okay to use in de-scrambling, since it is already known that there will be no remainder as the divisor was the multiplier in the scrambling process.

For both the scramble and de-scramble, the key is expected to remain the same for many frames and most likely for the whole content track, where a track can consist of a song or movie. Thus, with broadcasts, the key will probably be changing each track, and there are many ways to send keys for someone familiar with the state of the art in cryptology. When using PN sequences, as described below, the key for the PN sequence is the generator function, and it does not change for each MP3 song, i.e. defined as a track. Note that the generator function creates a random sequence that is identical each time, and is well know in the state of the art of cryptology. Alternatively, every content track, such as a song, may use one or more keys from a limited global list.

Fig. 34a displays the pseudo-code for an example of the scrambling or de-scrambling process. In this example, the content contains frames that begin with a sync code, and a header exists for each frame. Since the inverse of the XOR function is itself, the pseudo-code for the scrambling and de-scrambling process is identical.

Content scrambled or de-scrambled by this simple example could include MPEG audio data, such as Layer III (MP3) or AAC. MPEG audio's sync code is '1111 1111 1111'. The advantage of such approaches are numerous. For example, portable players can quickly display information about the song's length, artist, resolution, etc., before a user decides to play the song. Likewise, the header may contain copyright information that the player is required to check before playing.

The process begins at the beginning of the content (box **200**). Then, the sync code is found, usually being the first few bits of the content (box **205**). Next, the header data is skipped, possibly reading its own size from data after the sync code (box **210**). Then, the M content bits for that frame are scrambled using an XOR operation with the M content bits and M bits of a PN sequence (box **215**). Fig 34b shows the input and output for the XOR function. Next, the content is checked to see if another frame exists (box **220**). If another frame exists, the process continues at box **205** where the next sync code is located. Usually, the size of the frame can be read from the frame's header, which aids in searching for the next sync code. If no content remains, the process is complete (box **225**).

In this example, the size of M determines the robustness to brute-force attack, where the attacker's purpose is to obtain the original content. The larger M, the more robust the scrambled content is to attack. However, the smaller M, the more efficient the scrambling and de-scrambling processes can be. M can be any number which is both greater than the number of bits that can be repaired by error correction, and less than the number of non-header content bits for that frame.

The location of the M bits, which are scrambled in the frame, must be known and contain bits that are critical to the content's integrity. They could be the M bits after the header. However, in MP3 data, the frame data may not start after the header. In this case, the scrambled bits could be the first M

bits of the data for that frame. These bits determine the allocation of audio bits and are critical to the integrity of the file.

Fig. 35 shows hardware suitable for use in implementing the scrambling or de-scrambling process. The hardware includes a digital logic processor **300** and a digital memory **310**. The logic processor performs the calculations and logic for this process. The logic processor **300** may be defined as the equivalent of a digital signal processor (DSP), general-purpose central processing unit (CPU), a specialized CPU, including media processors, or application specific circuitry (ASIC). A likely DSP chip is one of the Texas Instruments TMS320 product line. A CPU could include one of Intel's Pentium line or Motorola/IBM's PowerPC product line. The ASIC can easily be designed by someone familiar with the state of the art and the above pseudo-code and description. The design of code for controlling logic processor **300** is also simple for someone familiar with the state of the art given the above pseudo-code and description. The memory **310** includes RAM when using a digital processor, and is used to store the program and other necessary variables.

## Conclusion

Having described and illustrated the principles of my technology with reference to various embodiments, it should be apparent that the technology can be modified in arrangement and detail without departing from such principles.

For example, while many of the embodiments employed watermark technology to identify the object or content, this may not be essential. Other marking techniques may be employed in the appropriate circumstances.

Similarly, while certain of the processes are detailed as being performed at a certain location relative to the user, the location of such processes is not generally critical. That is, tasks can be allocated among processing devices as best befits the context (provided security issues are adequately addressed).

While reference has occasionally been made to applications involving images and video, the focus on exemplary audio applications may obscure this fact. Thus, it should be emphasized that the techniques detailed above are equally applicable to other forms of media beyond audio.

While the detailed embodiment was described as changing the values of single samples, in other embodiments it may be desirable to change the values of plural neighboring samples, e.g., to increase durability of the watermark in the presence of corruption.

Similarly, while the detailed embodiments were described as embedding auxiliary data into content, the form of representation of the auxiliary was not much discussed. In some embodiments a payload of N bits may be encoded as M bits, where M>N (i.e. with partial or complete redundancy). The redundancy can include repetition of the N bits payload through the content; BCH-, convolutional-, turbo-, etc-coding of the N-bits to provide robustness and/or error correction; CRC or ECC codes, etc.

While the detailed embodiments are systems of many parts, it will be recognized that novelty also resides in individual components thereof, and that such components can also be employed in other systems and devices.

The particular combinations of elements and features in the above-detailed embodiments are exemplary only; the interchanging and substitution of these teachings with those of other embodiments and with the teachings of the incorporated-by-reference documents is also contemplated.

In view of the wide variety of embodiments to which the principles and features discussed above can be applied, it should be apparent that the detailed technology is illustrative only and should not be taken as limiting the scope of the invention. Rather, I claim as my invention all such modifications as may come within the scope and spirit of the following claims and equivalents thereof.

**I Claim:**

1. A method of steganographically encoding plural-bit auxiliary data within content, the content comprising audio, video, or still imagery, the content comprising plural samples, each having a value, the method including changing at least certain samples of the content to thereby encode a certain bit of the auxiliary data therein, said encoding being accomplished without transforming the content to a complementary domain, the method characterized by detecting a predetermined feature in the content, identifying a sample corresponding to said feature, and changing said identified sample, or a sample within a specified sample neighborhood of said identified sample, to effect said encoding.

2. The method of claim 1 in which the feature is an attribute associated with a neighborhood of adjoining samples.

3. The method of claim 1 in which the changing comprises changing in accordance with said bit and an original value of the sample being changed.

4. The method of claim 1 in which the changing comprises changing in accordance with said bit and so that the changed value has a specified relationship with neighboring sample values.

5. The method of claim 1 in which the predetermined feature is a sample having a value exceeding a first threshold, and wherein a difference between said sample and a neighboring sample has a value within a predetermined threshold.

6. The method of claim 5 in which said neighboring sample is an adjoining sample.

7. The method of claim 1 in which the changing spectrally spreads the energy of the auxiliary data.

8. The method of claim 1 that includes changing a sample adjoining said identified sample.

9. A content protection system substantially as described and illustrated above.

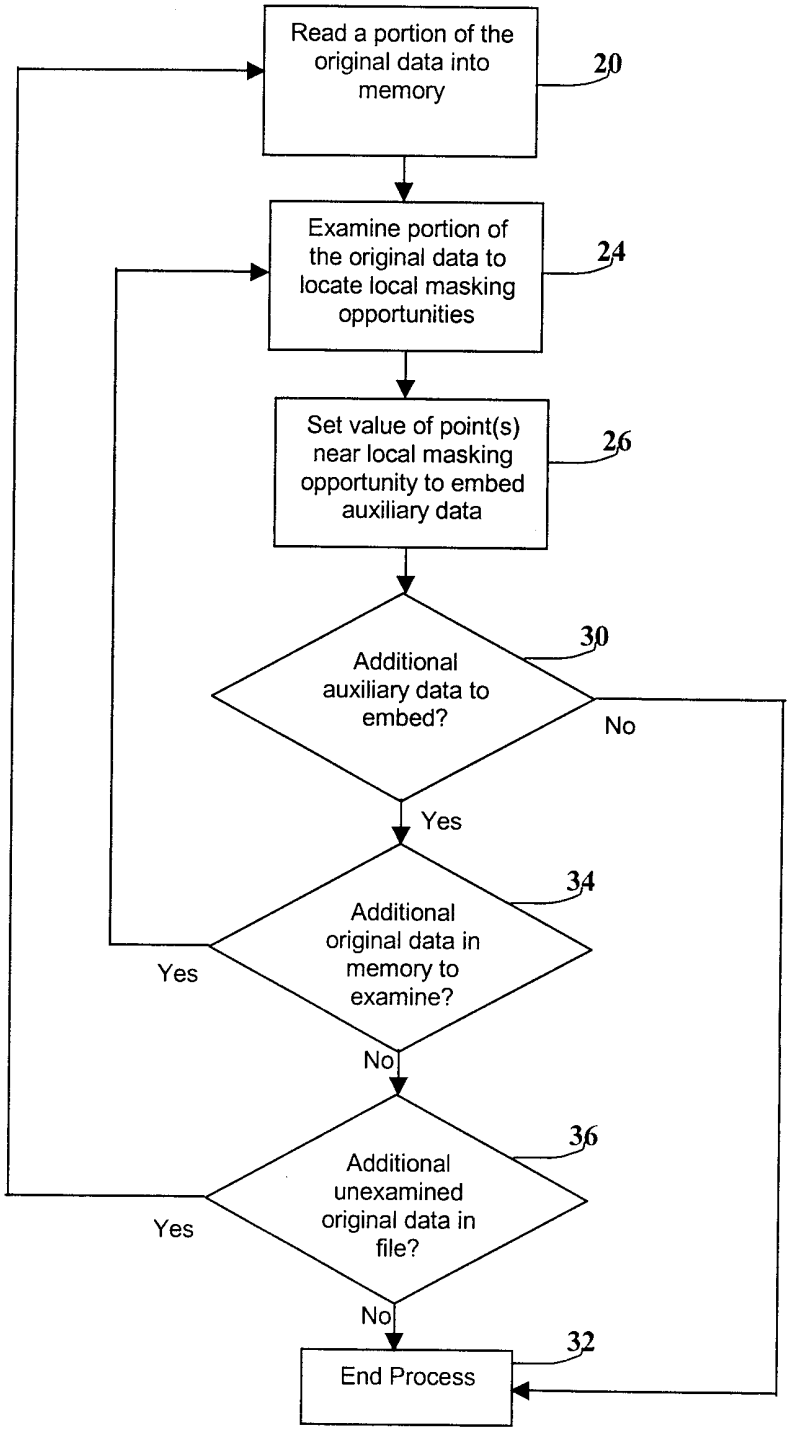

Read a portion of the original data into memory
20
Examine portion of the original data to locate local masking opportunities
24
Set value of point(s) near local masking opportunity to embed auxiliary data
26
Additional auxiliary data to embed?
30
No
Yes
Additional original data in memory to examine?
34
Yes
No
Additional unexamined original data in file?
36
Yes
No
End Process
32

Fig 1

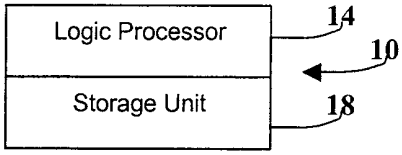



Fig. 2

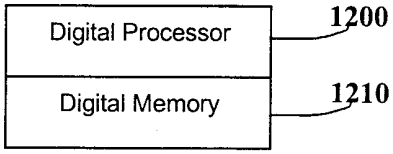



Fig. 12

| | | |
|---|---|---|
| A to D Converter | Sample and Hold | D to A Converter |
| Comparator | PLC | Delays |

Fig 13

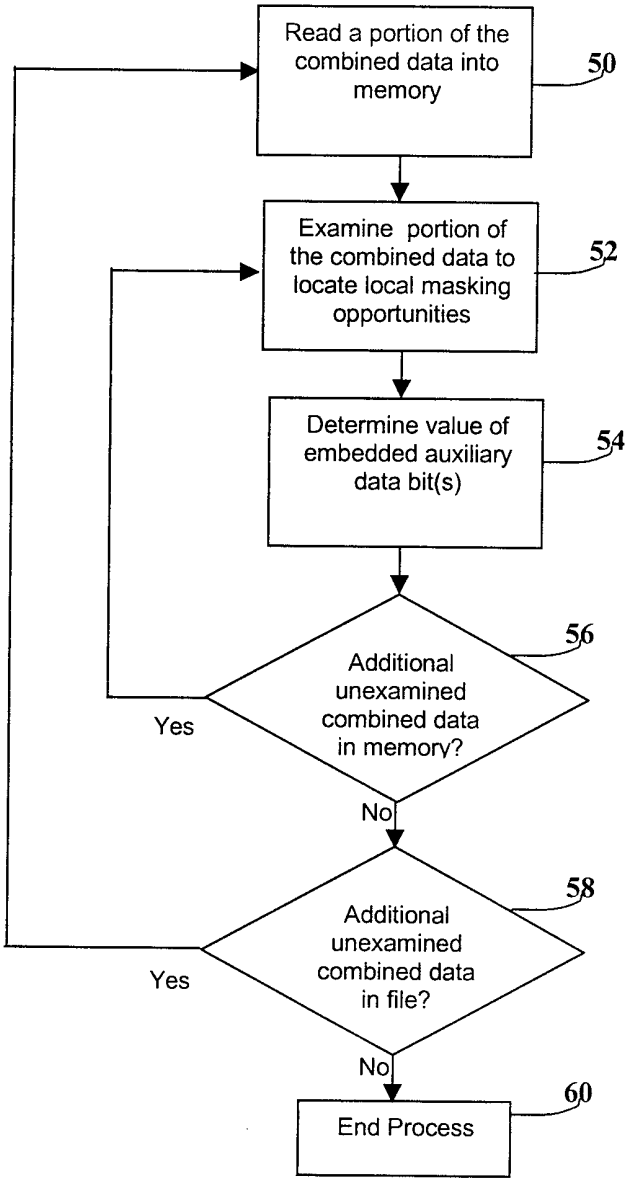

Read a portion of the combined data into memory
50
Examine portion of the combined data to locate local masking opportunities
52
Determine value of embedded auxiliary data bit(s)
54
Additional unexamined combined data in memory?
56
Yes
No
Additional unexamined combined data in file?
58
Yes
No
End Process
60

Fig 3

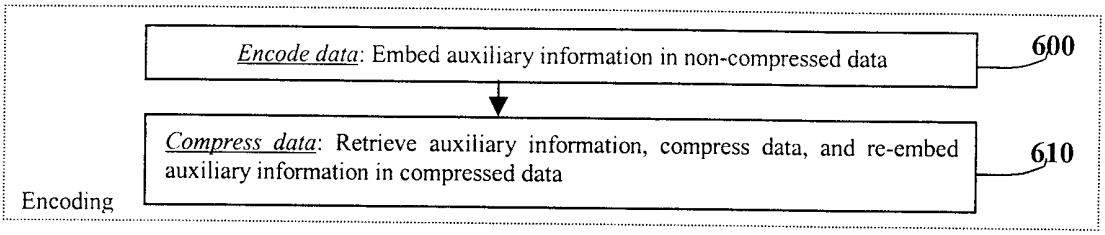



Fig. 10A

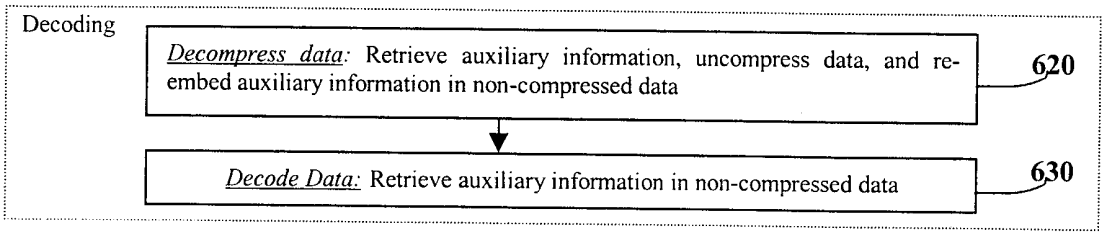



Fig. 10B

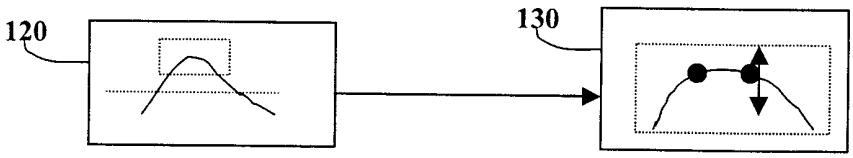



Fig. 4


Fig. 7

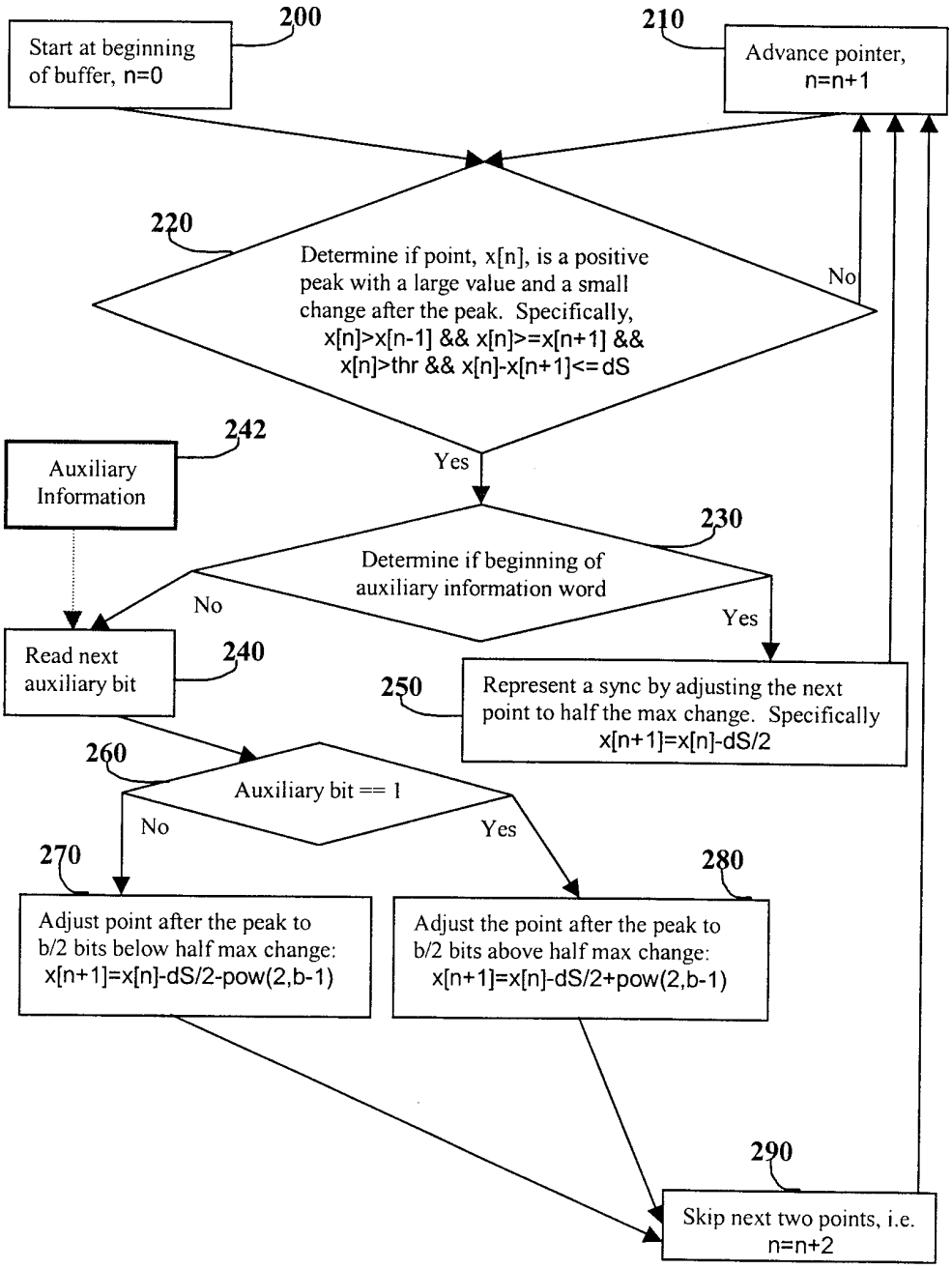

200
Start at beginning of buffer, n=0
210
Advance pointer, n=n+1
220
Determine if point, x[n], is a positive peak with a large value and a small change after the peak. Specifically, x[n]>x[n-1] && x[n]>=x[n+1] && x[n]>thr && x[n]-x[n+1]<=dS
No
Yes
242
Auxiliary Information
230
Determine if beginning of auxiliary information word
No
Yes
Read next auxiliary bit
240
250
Represent a sync by adjusting the next point to half the max change. Specifically x[n+1]=x[n]-dS/2
260
Auxiliary bit == 1
No
Yes
270
280
Adjust point after the peak to b/2 bits below half max change: x[n+1]=x[n]-dS/2-pow(2,b-1)
Adjust the point after the peak to b/2 bits above half max change: x[n+1]=x[n]-dS/2+pow(2,b-1)
290
Skip next two points, i.e. n=n+2

**Fig. 5**

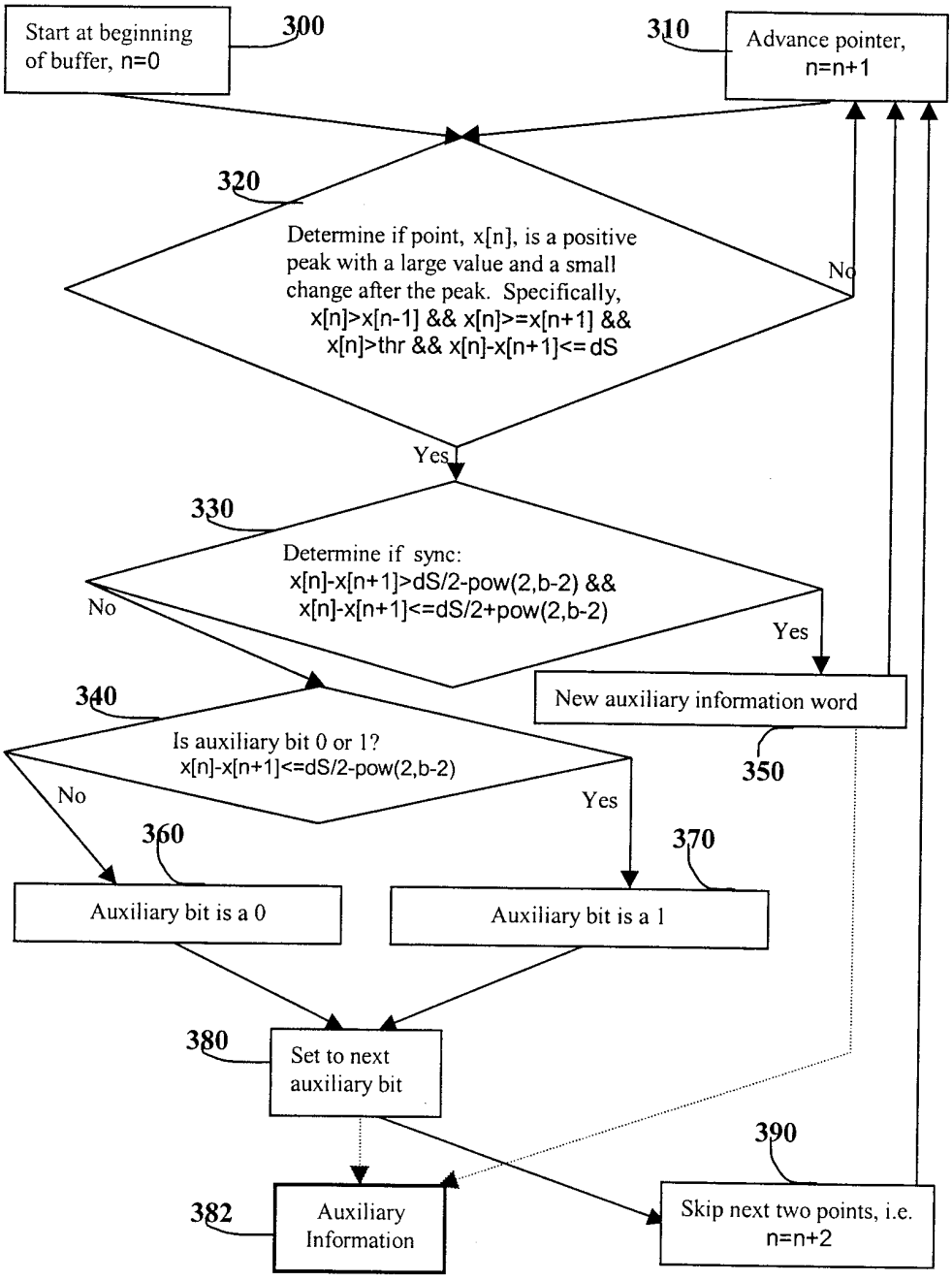

Start at beginning of buffer, n=0
300
310
Advance pointer, n=n+1
320
Determine if point, x[n], is a positive peak with a large value and a small change after the peak. Specifically, x[n]>x[n-1] && x[n]>=x[n+1] && x[n]>thr && x[n]-x[n+1]<= dS
No
Yes
330
Determine if sync: x[n]-x[n+1]>dS/2-pow(2,b-2) && x[n]-x[n+1]<=dS/2+pow(2,b-2)
No
Yes
New auxiliary information word
350
340
Is auxiliary bit 0 or 1? x[n]-x[n+1]<=dS/2-pow(2,b-2)
No
Yes
360
370
Auxiliary bit is a 0
Auxiliary bit is a 1
380
Set to next auxiliary bit
382
Auxiliary Information
390
Skip next two points, i.e. n=n+2

**Fig. 6**

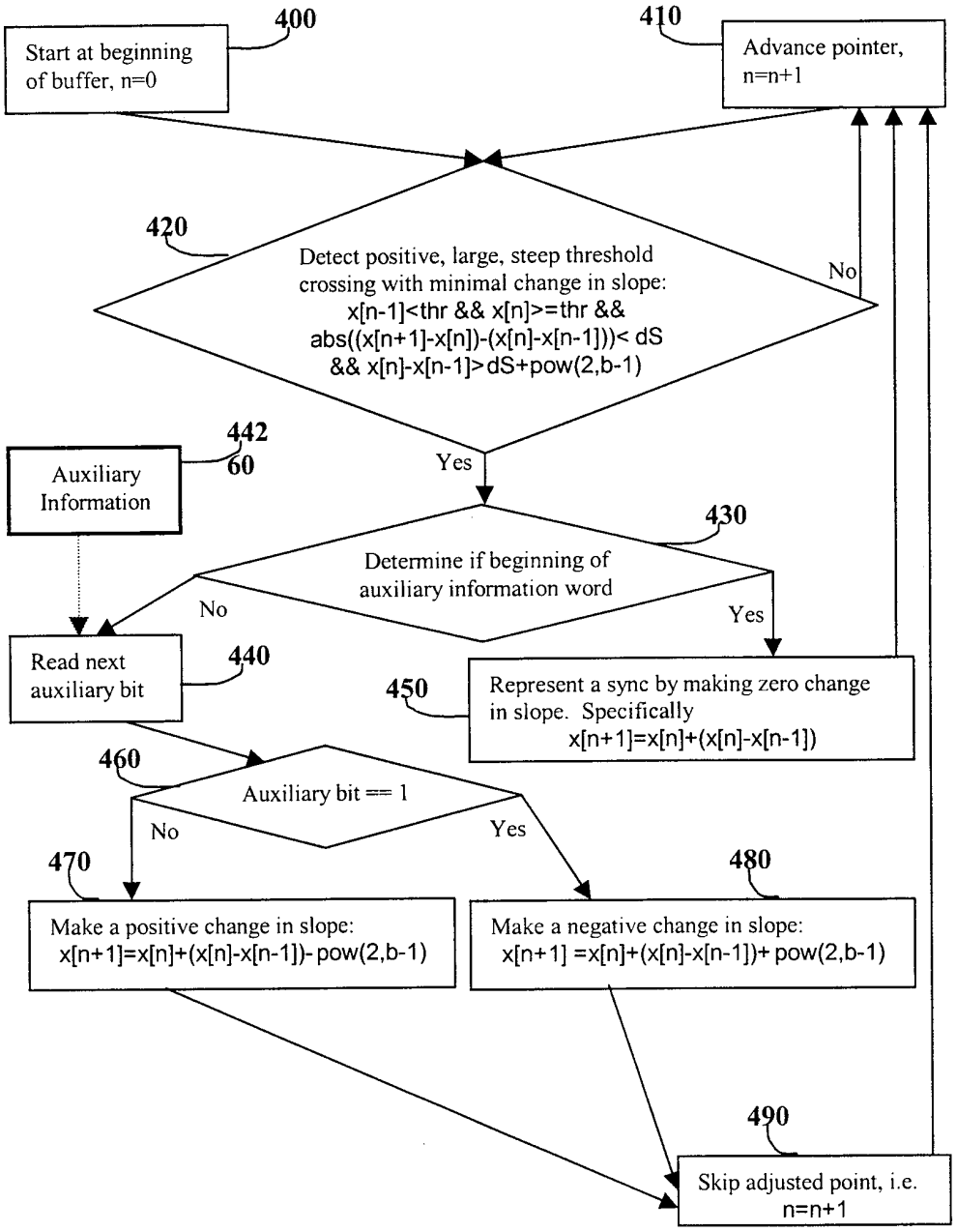

400
Start at beginning of buffer, n=0
410
Advance pointer, n=n+1
420
Detect positive, large, steep threshold crossing with minimal change in slope:
x[n-1]<thr && x[n]>=thr &&
abs((x[n+1]-x[n])-(x[n]-x[n-1]))< dS
&& x[n]-x[n-1]>dS+pow(2,b-1)
No
Yes
442
60
Auxiliary Information
430
Determine if beginning of auxiliary information word
No
Yes
Read next auxiliary bit
440
450
Represent a sync by making zero change in slope. Specifically
x[n+1]=x[n]+(x[n]-x[n-1])
460
Auxiliary bit == 1
No
Yes
470
480
Make a positive change in slope:
x[n+1]=x[n]+(x[n]-x[n-1])- pow(2,b-1)
Make a negative change in slope:
x[n+1] =x[n]+(x[n]-x[n-1])+ pow(2,b-1)
490
Skip adjusted point, i.e. n=n+1

**Fig. 8**

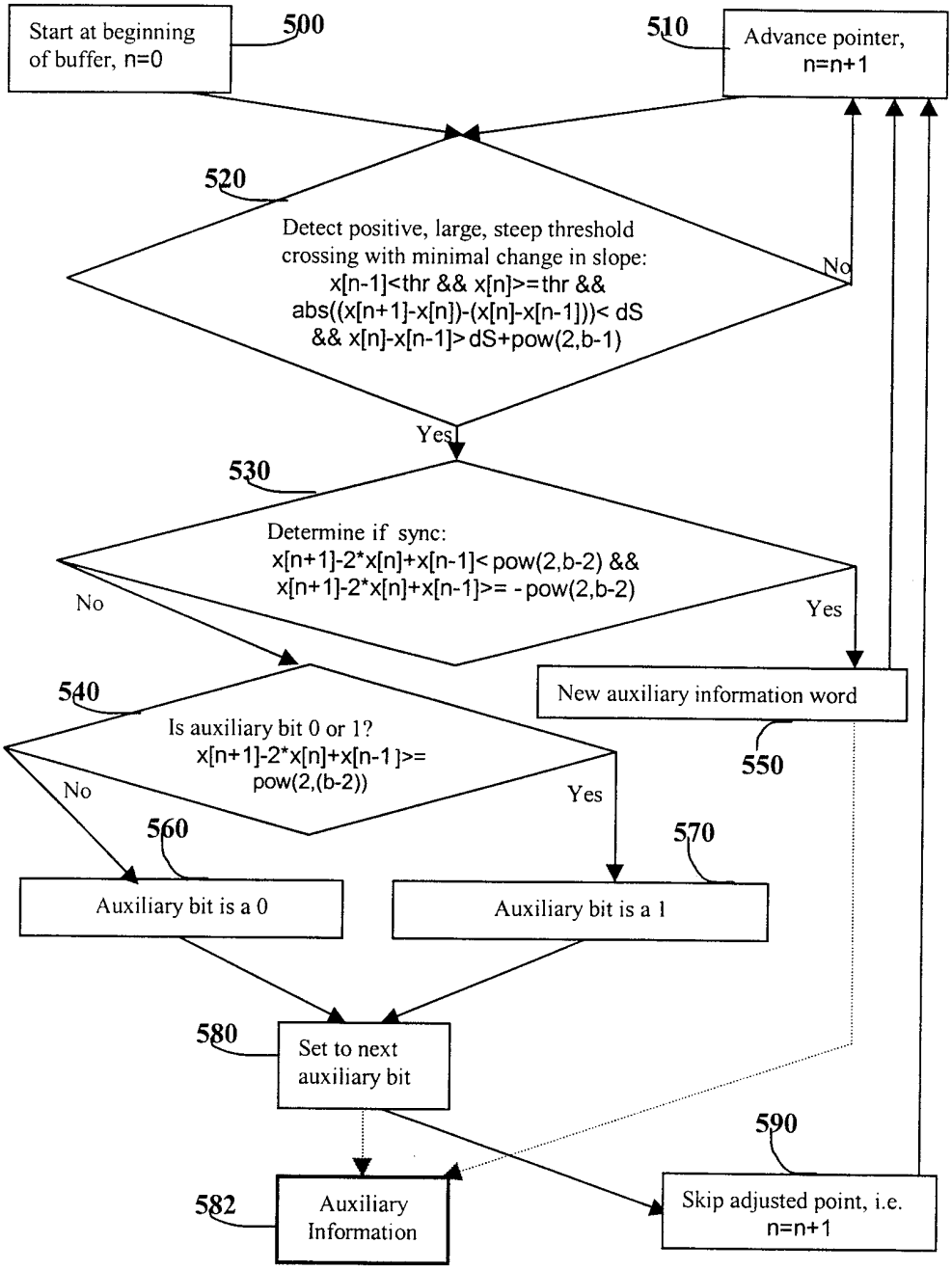

Start at beginning of buffer, n=0
500
510
Advance pointer, n=n+1
520
Detect positive, large, steep threshold crossing with minimal change in slope:
x[n-1]<thr && x[n]>=thr &&
abs((x[n+1]-x[n])-(x[n]-x[n-1]))< dS
&& x[n]-x[n-1]> dS+pow(2,b-1)
No
Yes
530
Determine if sync:
x[n+1]-2*x[n]+x[n-1]< pow(2,b-2) &&
x[n+1]-2*x[n]+x[n-1]>= -pow(2,b-2)
No
Yes
New auxiliary information word
550
540
Is auxiliary bit 0 or 1?
x[n+1]-2*x[n]+x[n-1]>=
pow(2,(b-2))
No
Yes
560
570
Auxiliary bit is a 0
Auxiliary bit is a 1
580
Set to next auxiliary bit
590
582
Auxiliary Information
Skip adjusted point, i.e. n=n+1

Fig. 9

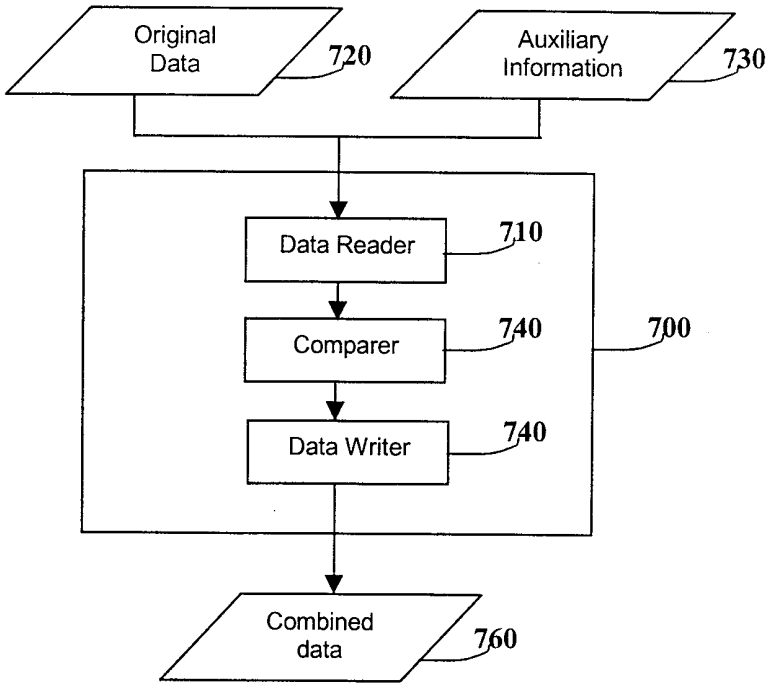



Fig. 11A

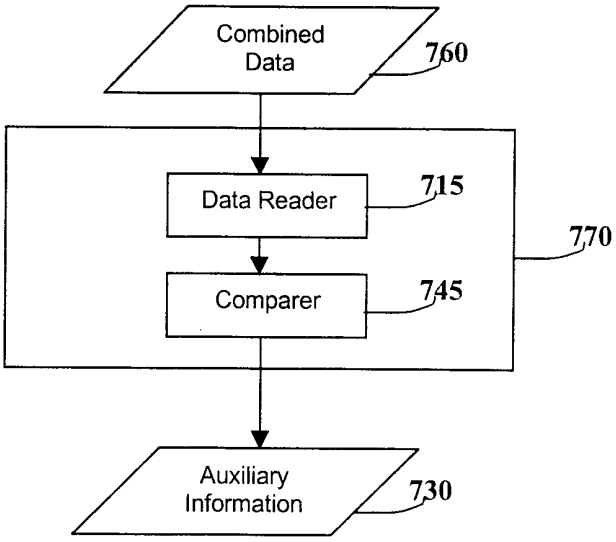



Fig. 11B

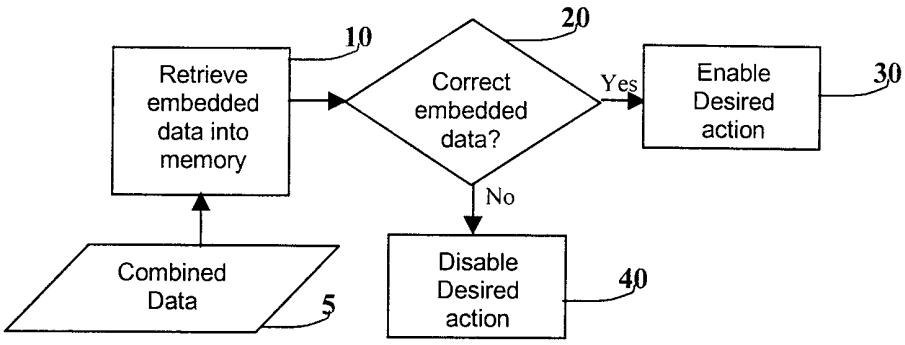



Fig. 14

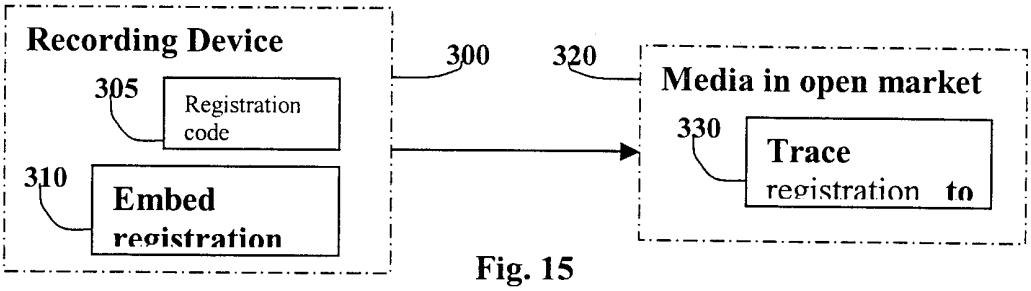



Fig. 15

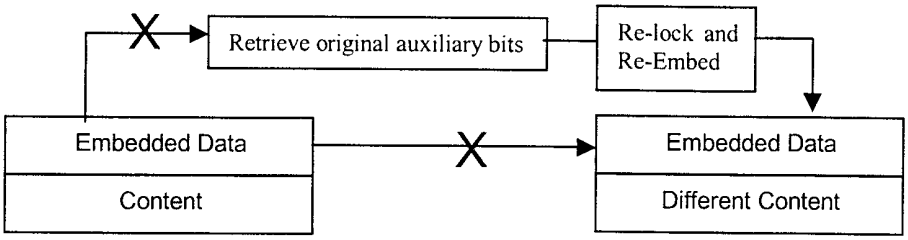



Fig. 16

XOR

| d | c | f(d,c) |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

Fig. 17

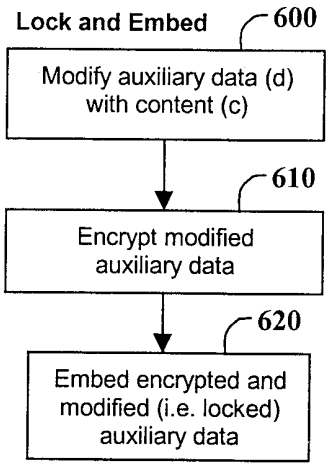



Fig. 18A

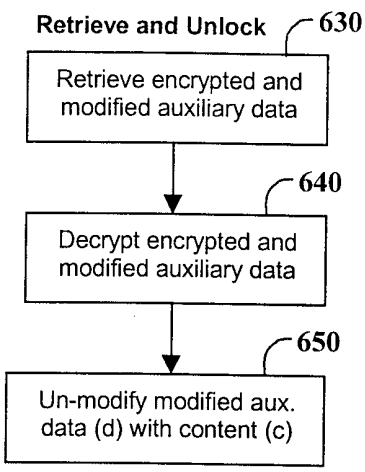



Fig. 18B

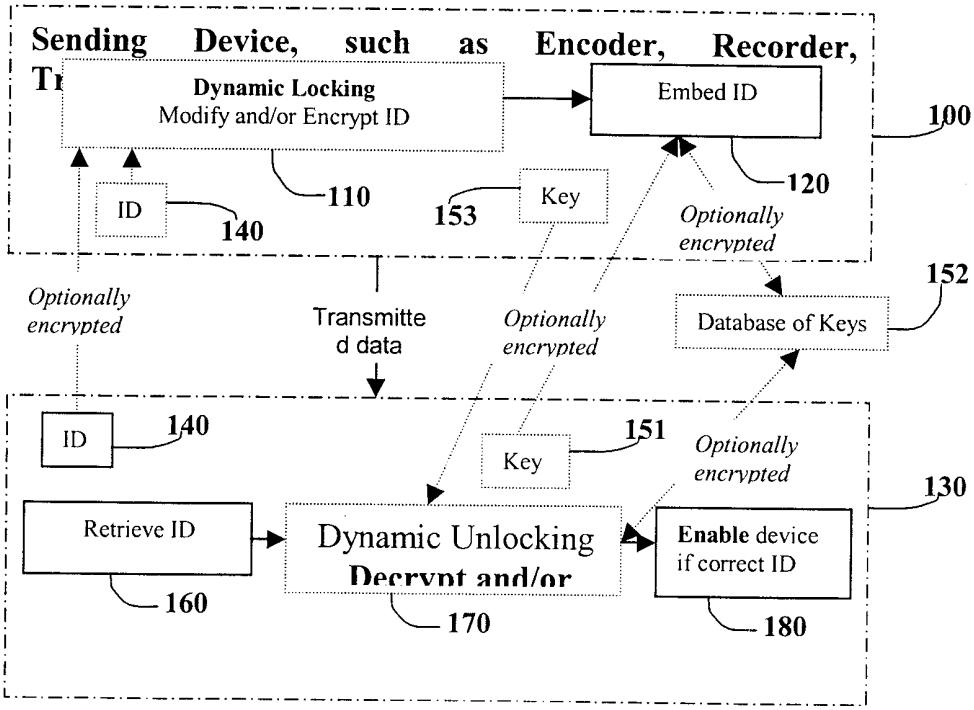



Fig. 21

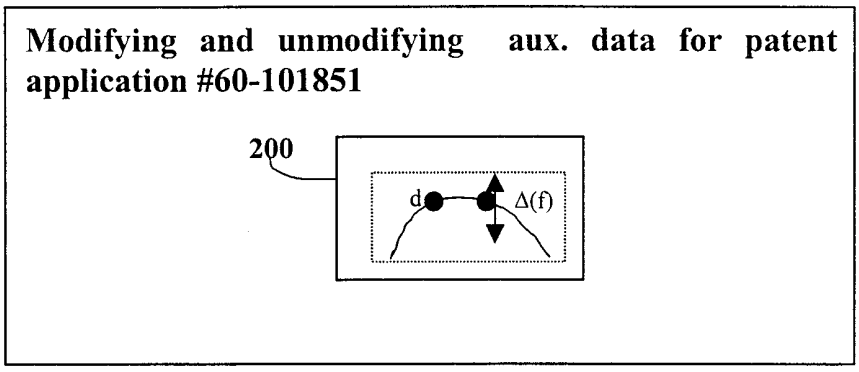



Fig. 19A

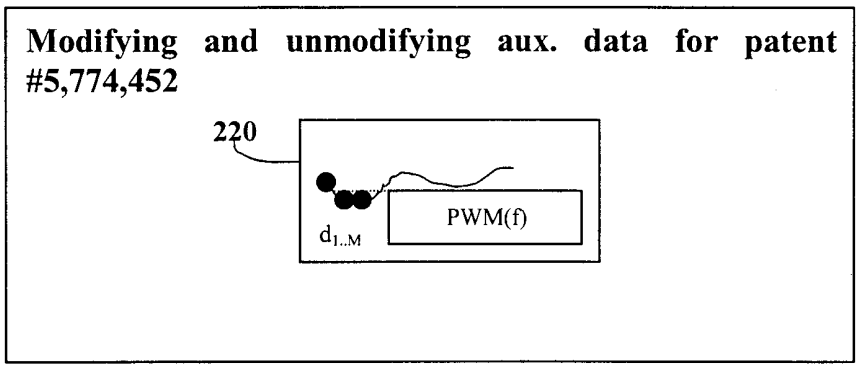



Fig. 19B

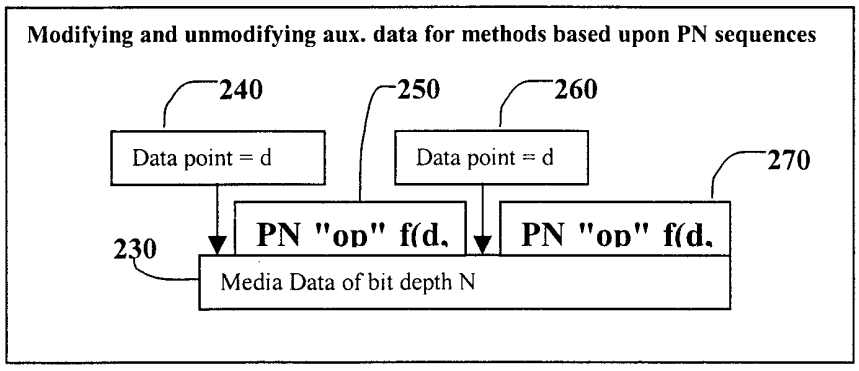



Fig. 19C

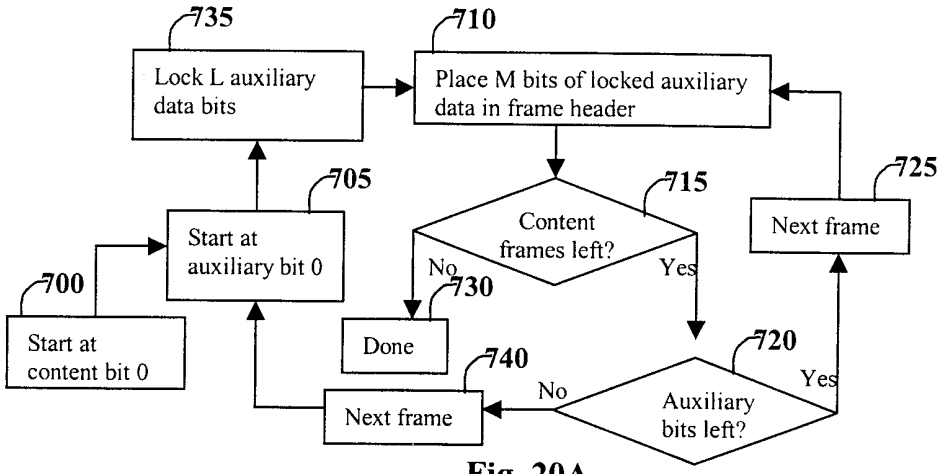



Fig. 20A

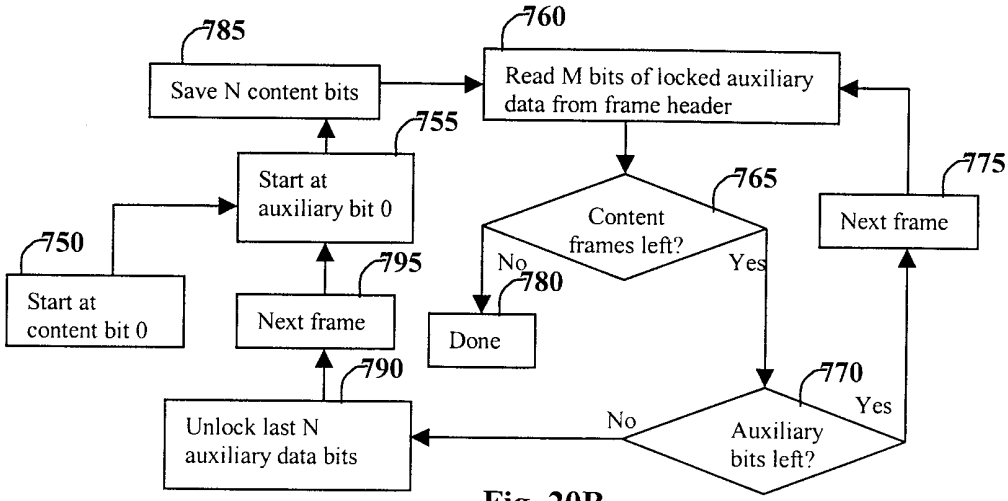



Fig. 20B

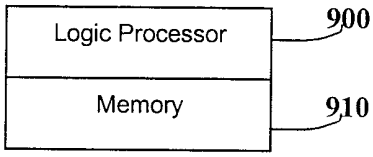



Fig. 22

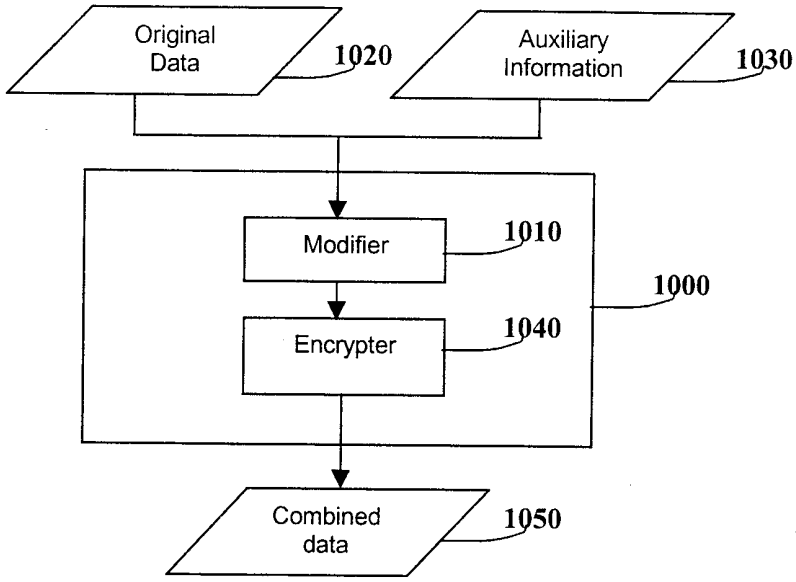



Fig. 23A

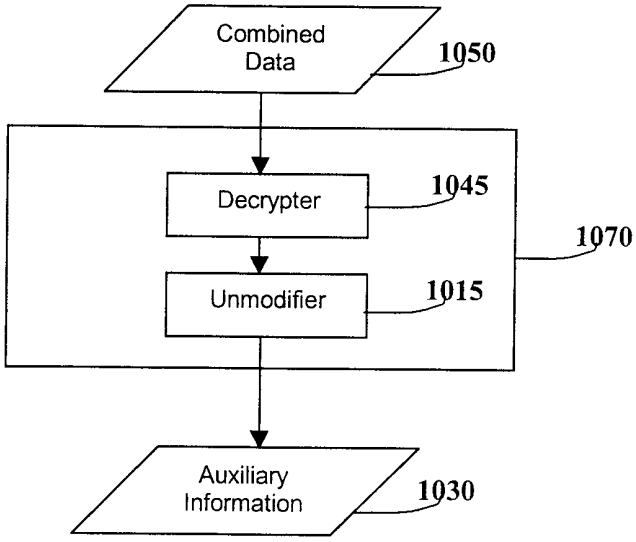



Fig. 23B

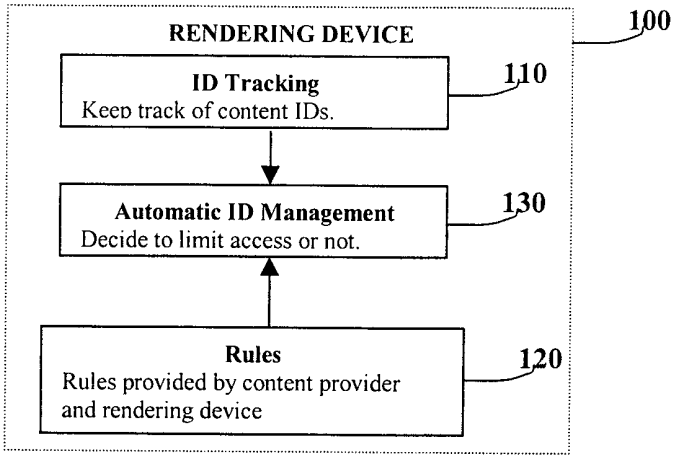



Fig. 25

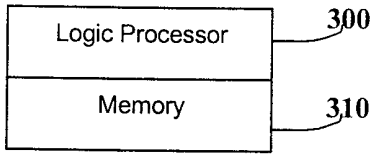



Fig. 27

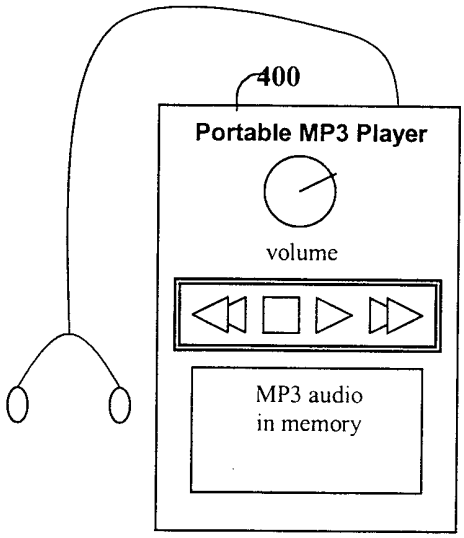



Fig 28

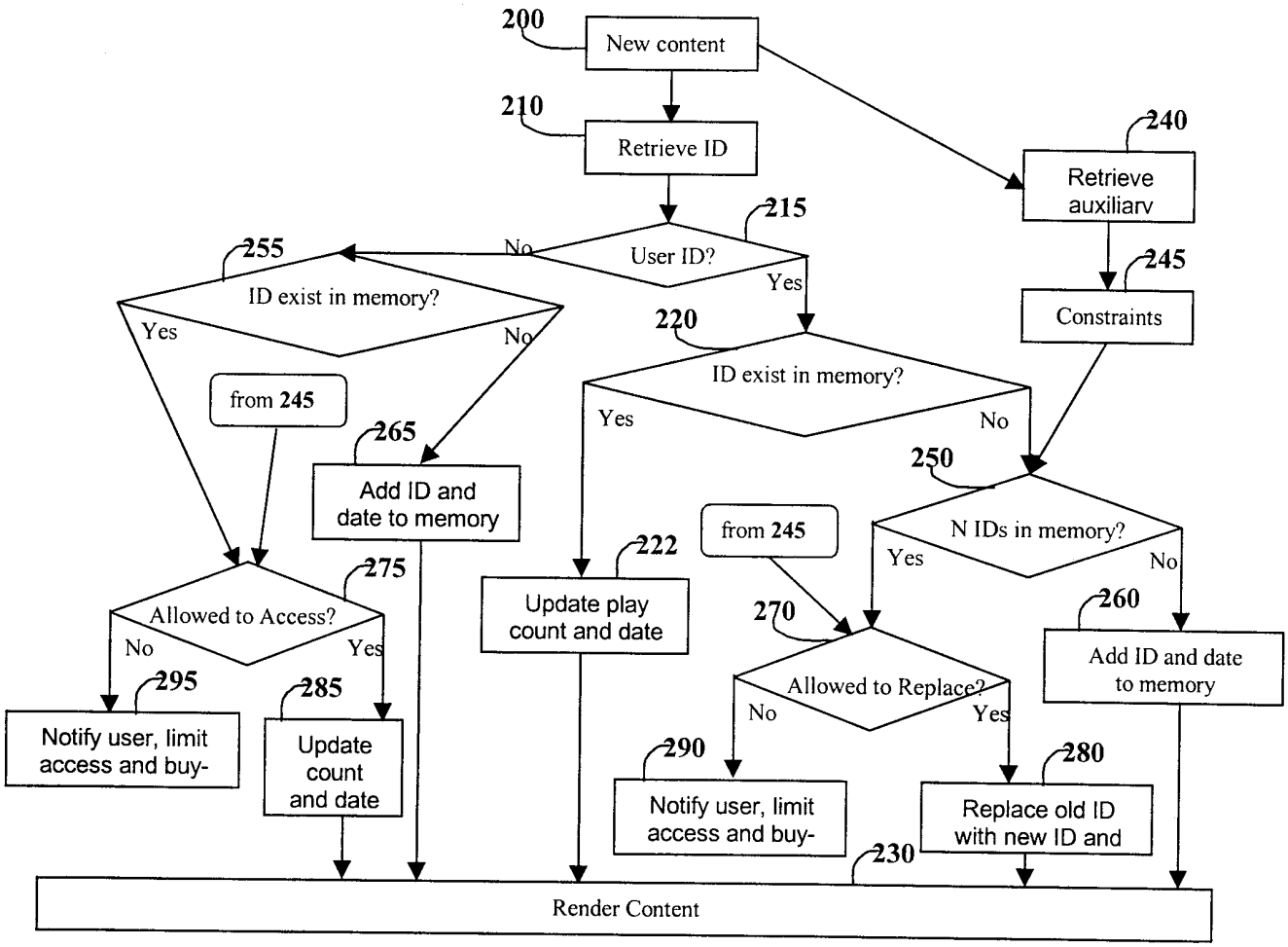

200
New content
210
Retrieve ID
240
Retrieve auxiliary
215
User ID?
No
Yes
255
ID exist in memory?
Yes
No
245
Constraints
220
ID exist in memory?
Yes
No
from 245
265
Add ID and date to memory
250
N IDs in memory?
Yes
No
from 245
222
Update play count and date
275
Allowed to Access?
No
Yes
270
Allowed to Replace?
No
Yes
260
Add ID and date to memory
295
Notify user, limit access and buy-
285
Update count and date
290
Notify user, limit access and buy-
280
Replace old ID with new ID and
230
Render Content

Fig. 26

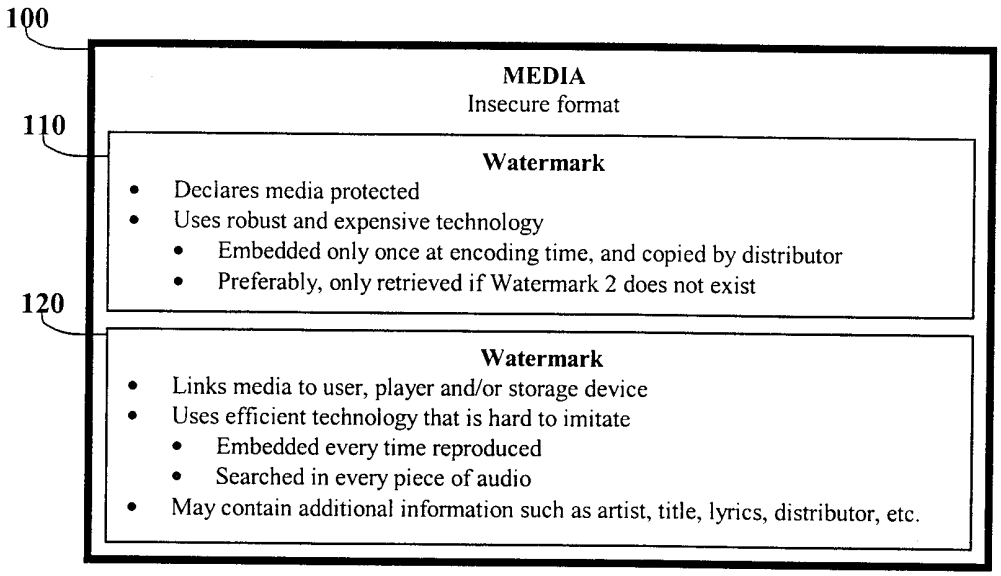



Fig. 29

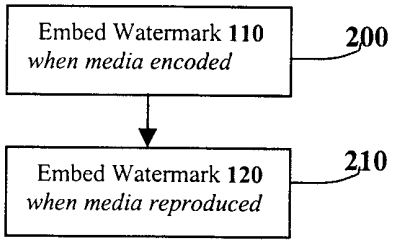



Fig. 30

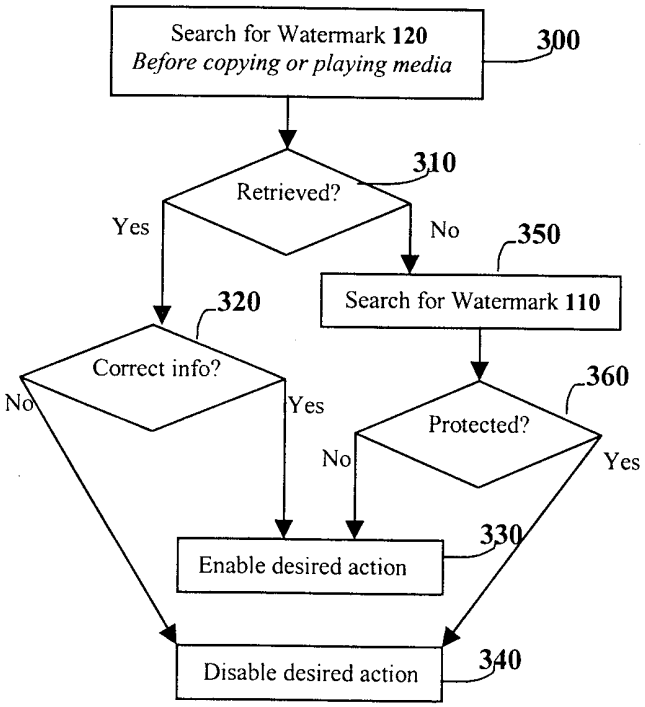



Fig. 31

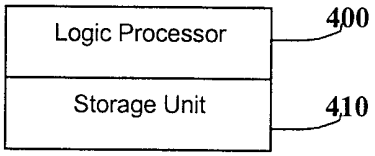



Fig. 32

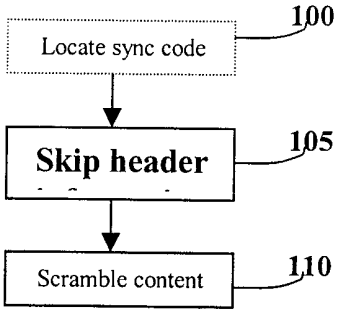



Fig. 33a

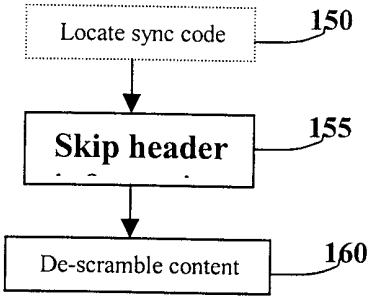



Fig. 33b

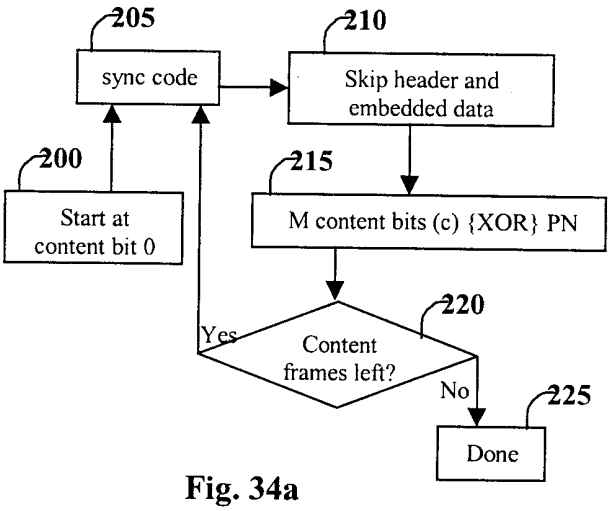



Fig. 34a

XOR

| c | PN | f(c,PN) |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

Fig. 34b

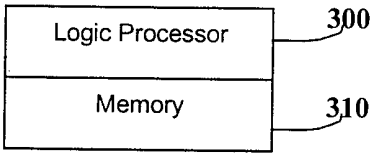



Fig. 35

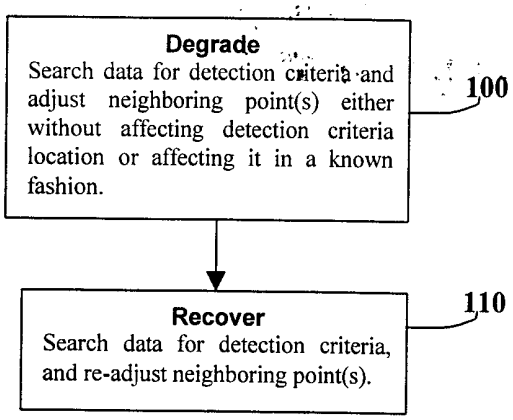



Fig. 36

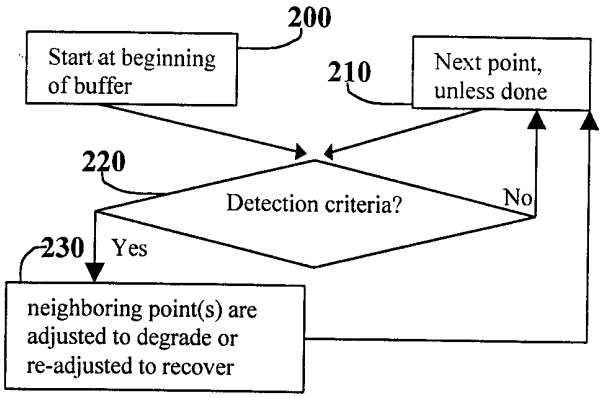



Fig. 37

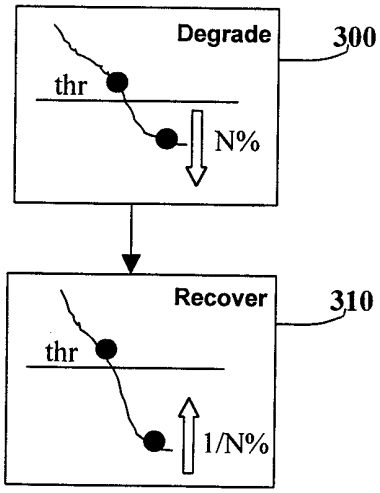



Fig. 38

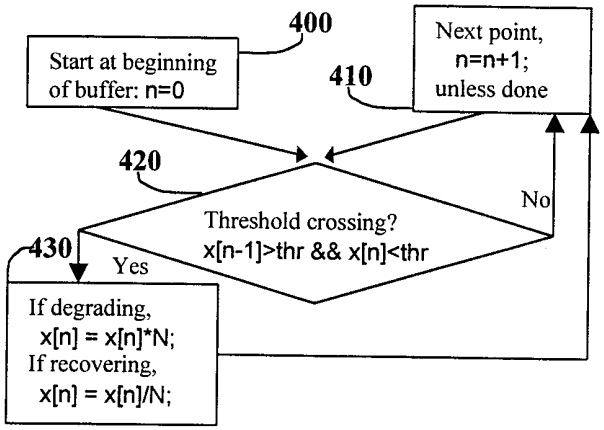



Fig. 39

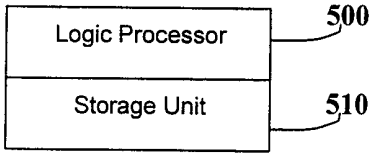



Fig. 40

# INTERNATIONAL SEARCH REPORT

| International application No. |
|---|
| PCT/US00/06296 |

**A. CLASSIFICATION OF SUBJECT MATTER**

IPC(6) : H04L 9/00
US CL : 713/168, 176

According to International Patent Classification (IPC) or to both national classification and IPC

**B. FIELDS SEARCHED**

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 713/168, 176

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
East, West

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

| Category * | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| X, P<br>---<br>Y, P | US 6,021,196 A (SANDFORD, II et al) 01 February 2000 (01.02.2000), abstract, columns 2-3, lines 62-33. | 1-3, 8-9<br>---<br>7 |
| X<br>---<br>Y | US 5,721,788 A (POWELL et al) 24 February 1998 (24.02.1998), abstract, figures 2A, 2, 3, and 5, column 4, lines 13-39. | 1-6, 8-9<br>---<br>7 |
| Y, P | US 5,930,369 A (COX et al) 27 July 1999 (27.08.1999), abstract, column 7, lines 16-37. | 7 |
| A, E | US 6,061,793 A (TEWFIK et al) 09 May 2000 (09.05.2000), ALL. | 1-9 |
| A, E | US 6,049,627 A (BECKER et al) 11 April 2000 (11.04.2000), ALL. | 1-9 |
| A, E | US 6,044,182 A (DALY et al) 28 MARCH 2000 (28.03.2000), ALL. | 1-9 |
| A, P | US 5,943,422 A (VAN WIE et al) 24 AUGUST 1999 (24.08.1999), ALL. | 1-9 |
| A, P | US 5,912,972 A (BARTON) 15 JUNE 1999 (15.06.1999), ALL. | 1-9 |
| A | US 5,875,249 A (MINTZER et al) 23 FEBRUARY 1999 (23.02.1999), ALL. | 1-9 |

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

| Date of the actual completion of the international search | Date of mailing of the international search report |
|---|---|
| 30 May 2000 (30.05.2000) | 05 JUL 2000 |
| Name and mailing address of the ISA/US<br>Commissioner of Patents and Trademarks<br>Box PCT<br>Washington, D.C. 20231<br>Facsimile No. (703)305-3230 | Authorized officer<br>Tod Swann For Eugenia Zagyan<br>Telephone No. (703) 305-3900 |

Form PCT/ISA/210 (second sheet) (July 1998)