Protection on Audio CDs

Romain Ayala
Ensimag, Grenoble(France)
Romain.Ayala@ensimag.imag.fr

Laura Medji
Ensimag, Grenoble(France)
Laura.Medji@ensimag.imag.fr

Terence Momo
Ensimag, Grenoble (France)
Terence.Momo@ensimag.imag.fr

ABSTRACT

During this project we tried to find an effective way to protect audio CDs. To do this, we have experienced and compared several methods from the technology known as SCRATCH or insertion of wrong data. It consists, either by a manual way or by a program, in making the copy impossible (difficult) by inserting erroneous bytes on the CD. Among all the possibilities we have chosen two of them. The first one is a manual way that consists in scratching the CDs. In the other method, we modify the matrix representing the encoded audio data (the physical 32 bytes-frames encoded by the EFM modulation on the CD). Both methods are based on the two different ways to read audio CD's: the analogical way (in a classic audio player) with the temporal interpolation and the numerical way (in a CD-Rom driver that also has the analogical mode).

KEYWORDS

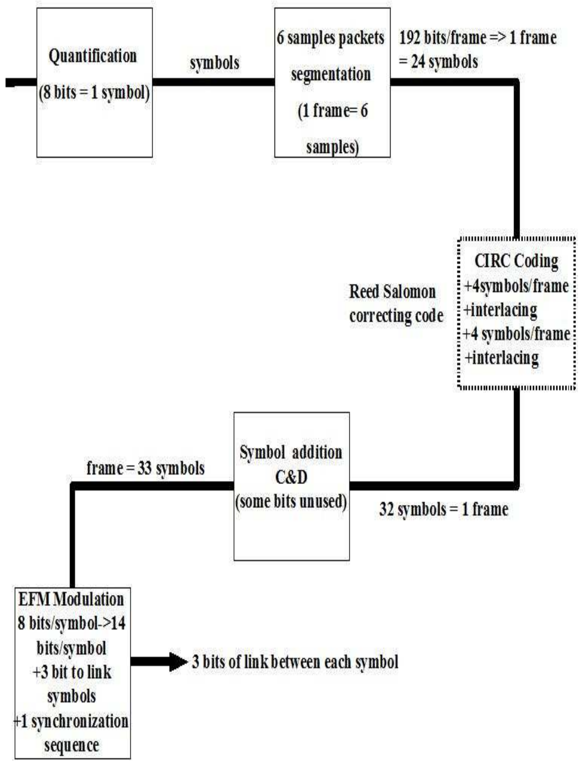
Interpolation, CIRC coding, Table of Content (TOC), Digital Rights Management (DRM), Frame, Sector, Scratch.

1. INTRODUCTION

At the age of Mp3s and pirated copies, the protection of CDs is, more than ever, a challenge for many retailers. That is why we have seen, for some years, the apparition of many effective protection technologies for CDs. These protections can be classified into three categories. The first one is the "SCRATCH" or data codeword modifications where we insert a certain number of erroneous bits. The second is the modification of the TOC which manipulates the tracks length and the information format. The latest, largest, is called DRM. It aims at controlling all the possible user utilizations of the product. To do this it uses the principles described above but also other software means.

2. CONTEXT

CDs encoding is a long and complex process. We try to explain it on the following schema:


Quantification
(8 bits = 1 symbol)
symbols
6 samples packets segmentation (1 frame= 6 samples)
192 bits/frame => 1 frame = 24 symbols
CIRC Coding
+4symbols/frame
+interlacing
+4 symbols/frame
+interlacing
Reed Salomon correcting code
32 symbols = 1 frame
Symbol addition C&D (some bits unused)
frame = 33 symbols
EFM Modulation 8 bits/symbol->14 bits/symbol +3 bit to link symbols +1 synchronization sequence
3 bits of link between each symbol

As we can see, the audio CDs encoding is made through five stages:

- During the quantification, bits representing audio data are transformed into symbols. Each symbol is made up of 8 bits. A sample is made up of 16 bits and there are two channels in the signal (stereo).

- Then, during the second step, symbols are transformed into frames. Each frame is composed of 24 symbols (6 samples on the left channel and 6 samples on the right one).

The two first steps are used for creating words of 24 symbols (=bytes) which are necessary to be processed by the CIRC coding. We are now going to focus on that specific step where our algorithms intervene.

- The CIRC coding is based on several stages: first, a shortened Reed-Solomon coding; then, an interlacing with a delay of 4 and finally, another shortened Reed-Solomon coding. After the CIRC coding, the frames are made up of 32 bytes and data are represented by a matrix of 32 lines and (number of frames + 4*(28-1)) columns (due to the interlacing-4).

The next two stages are explained in the schema: first, one symbol of synchronization is added to each frame and then, frames are encoded with the EFM modulation that allows the lossless burning and reading of these frames on an optical support.

3. THEORETICAL ANALYSIS

3.1 Insertion of wrong data (scratch)

This method consists in making a physical scratch on the CD in order to corrupt the data and prevent the user from copying the CD without any error. It can be seen as a type of protection since the true data are not recovered.
Theoretically, the CIRC coding enables a correction of 15 successive frames, which correspond to a length of 8820 bits on the track (2.5 mm) or 15*32 = 480 bytes of signal. Another interlacing is also used to make the temporal interpolation easier because the audio signal is much correlated. With this technique, we can recover 48 successive frames, which correspond to a length of 8.5 mm.

On the contrary of analogical players that use the interpolation, numerical players are more sensitive to errors in data because they process to a bit-per-bit reading. It is this characteristic that we want to make the most of by simulating these two reading modes.

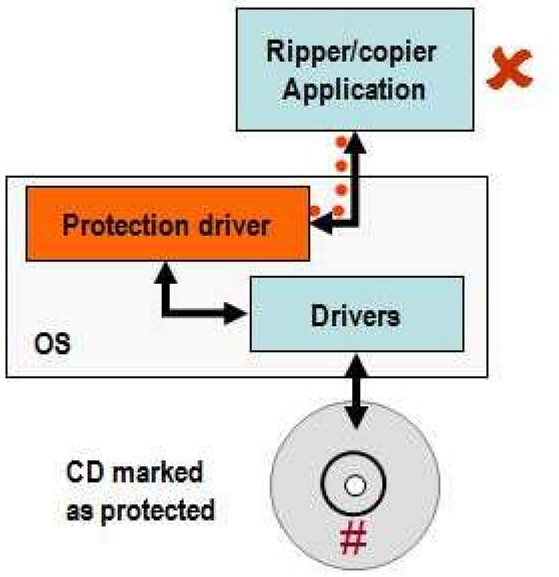
3.2 Invalid table of contents (TOC)

The TOC is the central index of every CD. It contains the start positions of the individual tracks on the CD. An audio CD player reads the TOC when the CD is inserted and then knows where each track starts and how long the CD is. The TOC is saved in the lead-in area of the CD. On a multisession CD in every lead-in of every session there is a new one.
To protect CDs thanks to the use of the TOC we need to create a multisession CD with a valid TOC on its first session and a wrong one on another one. Indeed, audio CD players only read the first TOC created on the CD whereas CD-Rom drivers use the latest. So, the user of CD-Rom drivers will listen to his audio data but will not be able to copy them.
To corrupt the TOC, two main techniques have been found. Both are based on the adding of a dummy track. On the one hand, as a CD cannot contain a track whose length is less than four seconds, we can make that track start two seconds before the end of the CD (lead-out start time). On the other hand, we can manipulate only the track modes. Indeed, a track has two possible modes: mode 1 and mode 2 and tracks from different modes cannot be copied on the same CD.

3.3 DRM

DRM regroups several different types of methods. Two of them have just been explained (scratch and the use of invalid table

of contents) so we will no longer focus on them. The one we are going to explain now is the technology that had been used by SONY BMG.
The following schema explains how CD DRM works:



The first time the protected CD is used, the autorun executes installer from the CD. The active protection driver is automatically installed between CD driver and application. So, even when the CD is no more inserted, the driver remains on the system.
Then when the user try to copy or rip a CD, the protection driver search for a watermark and, when it founds one, block the access to audio.

The SONY BMG technology consists in two things. It installs first a rootkit to conceal the software and then an aggressive protection software without the consumer consent. These methods jeopardized user security. Indeed the XCP rootkit and MediaMax aggressive installer used by SONY hide computer system files to the user and let him vulnerable to malicious attacks.

4. IMPLEMENTATION OF OUR PROTECTION: Insertion of errors during the coding process of the audio data

4.1 Presentation

After an intensive analysis, we have chosen to work on the method that seems the most efficient to us: the insertion of errors.

- First, we tried the manual way. We made a straight scratch of 5mm on a CD, which prevents the CD from being copied but does not make the reading impossible. Indeed, until 2.5mm scratched data can be recovered thanks to the CIRC coding. The interpolation allows a maximal scratch length of 8.5mm. So we had to choose a scratch between 2.5 and 8.5mm.
  Yet, this is an approximate method that may damage the data on the CD.

Thus, we decided to reproduce the scratching process through a C++ program with the objective of determining the number of successive frames that have to be corrupted in order to make the copy impossible without losing any audio data. Besides, the program contains other modification functions that enable a finer errors insertion. It can modify certain bytes of Reed-Solomon codeword to generate errors after the CIRC decoding.

- Our program operates on wave files. To begin, it extracts the data from the file and puts them byte per byte in a matrix that corresponds to the data after quantification. Each line of that matrix stands for a frame (24 bytes). Then, we make an interlacing which consists in mixing the 6 audio samples of every channel for each frame. After that, we do the CIRC coding of our matrix. After the encoding step, we select which modification we want to make to protect our data: on the one hand, we can simulate a single scratch in a specific sector of data; on the other hand, several scratches can be made. Moreover, we can modify some parts of the Reed-Solomon codeword in a finer way: arbitrarily, we modify the four bytes which stand for the redundancy of the RS codes C1 and C2 (we could modify the other bytes but these ones were easy to localize in the interlaced matrix). Finally, we are searching the best compromise between the noise added in the numerical reading and the quality of the data recovered after the interpolation in the analogical reading.

4.2 Detailed Explanations of our program

- *Extraction of the data*:

  We have chosen to operate on wave files because it is a simple format in which the audio data can be directly used. Indeed, the 44 first bytes of a wave file constitute the header and the data are found in the rest of the file. So, we put the header in a table to use it when we reconstruct the wave file at the end of the operation. Besides, the data are stocked in a matrix that will be the entry for the interlacing before C1 coding.

- *Interlacing before C1 coding*:

  This step aims at optimizing time Interpolation of audio data. In each frame, we mix the 6 audio samples of every channel. We sum up our process in the following schema:

A frame before the interlacing

G1	D1	G2	D2	G3	D3	G4	D4	G5	D5	G6	D6

The same frame after the interlacing

G1	G3	G5	D1	D3	D5	G2	G4	G6	D2	D4	D6

In this scheme, $(\mathrm{Gi};\ \mathrm{Di}\ )$ (with $\mathrm{i} \in [\![1;6]\!]$) represents an audio sample for a channel and a frame. Gi refers to the left byte of the sample and Di to the right one. The first and second 6 boxes of the table represent a channel.

- *CIRC Coding:*

  The CIRC coding is made up of three stages: the C1 coding, the interlacing phase and the C2 coding.

The C1 code is a shortened (28, 24) Reed-Solomon code. As we work with a (255, 251) Reed-Solomon code, we have to extend every frame to 251 bytes by filling the new bytes with zeros. Then, we apply a (255, 251) Reed-Solomon coding to each frame and we add the four calculated codes to the initial data frames. At the end of the C1 coding, we have a matrix whose lines are frames of 28 bytes and the four last bytes correspond to C1 codes as you can see on the next schema:

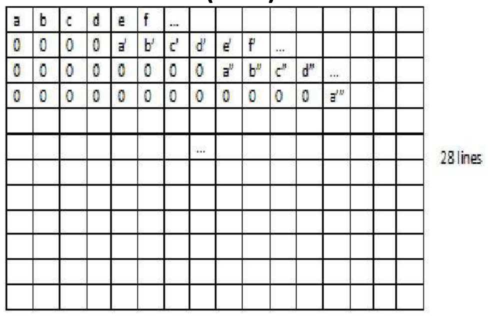
28 Bytes

B_1	B_2	B_3	……..	B_{23}	B_{24}	C1.1	C1.2	C1.3	C1.4

C1 Coding

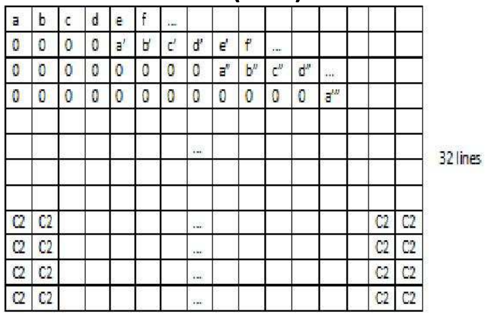
The latter matrix can now be interlaced in order to improve the capacity of recovering data in case of errors. The interlacing used in the CIRC coding has a delay of 4 and its principle is: we built a matrix where the j^th line is made up of the j^th byte of each frame in the order of the indexes and the j^th line is shifted to the right from j-4 bytes. The blanks of the interlacing matrix are filled with zeros. The following schema explains the principle:

Frames number + 4*(28-1) columns


a b c d e f …
0 0 0 0 a' b' c' d' e' f' …
0 0 0 0 0 0 0 0 a'' b'' c'' d'' …
0 0 0 0 0 0 0 0 0 0 0 0 a'''
…
28 lines

For the C2 coding, we operate on the interlacing matrix. The C2 code is a shortened (32, 28) Reed-Solomon code. The C2 coding principle is the same as the C1 coding one but it completes the columns of the interlacing matrix: the frames are made up of 32 bytes after the C2 coding as you can see on the following schema:

Frames number + 4*(28-1) columns



a	b	c	d	e	f	…									
0	0	0	0	a'	b'	c'	d'	e'	f'	…					
0	0	0	0	0	0	0	0	a''	b''	c''	d''	…			
0	0	0	0	0	0	0	0	0	0	0	0	a'''			
							…								
C2	C2						…							C2	C2
C2	C2						…							C2	C2
C2	C2						…							C2	C2
C2	C2						…							C2	C2

32 lines

- *Insertion of wrong bytes:*

  Errors are inserted in the data by three ways: the modification of columns of data (scratch method), the modification of C1 codes and the modification of C2 codes. We chose to change to zero the value of the bytes that undergo the modification.
  We introduce a few parameters to all those modifications such as the number of columns to modify per sector, the frequencies modification of columns and sectors.

- *CIRC Decoding:*

  Like the CIRC coding, the CIRC decoding is also made up of three stages: the C2 decoding, the reverse interlacing and the C1 decoding.
  The modified interlacing matrix is the entry of the C2 decoding that reduces the frames number of bytes from 32 to 28 bytes by eventually modifying those bytes in accordance with the C2 codes values.
  Then, we apply the reverse interlacing to the matrix obtained so as to remove the delay between the frames and to organize the frames in lines.
  Finally, each frame undergoes the C1 decoding and, at the end, we find back the data with eventual modifications.
  The C1 and C2 decoding are based on the (251, 254) shortened Reed-Solomon decoding. So, frames have to be extended before each decoding.
  The Reed-Solomon decoding consists in the arithmetic of syndromes and the Berlekamp algorithm that helps to find the error location polynomial.

- *Reverse Interlacing after decoding C1:*

  After the decoding of C1, we make the inverse process of interlacing before coding C1. We treat 6 audio samples of every channel in each frame just like before. We implement the algorithm inverse and find again our frame like shown in the following scheme:

A frame before the reverse interlacing

G1	G3	G5	D1	D3	D5	G2	G4	G6	D2	D4	D6

The same frame after the reverse interlacing

G1	D1	G2	D2	G3	D3	G4	D4	G5	D5	G6	D6

  This interlacing enables not to damage successive samples in case of a big scratch. If an error on a sample is detected, we can solve it by making the average with the previous sample and the next one: this is the temporal interpolation.

- *Reconstruction of the wave files*:
  At the end, we have the matrix whose elements are the audio data of the new corrupted wave file. We just have to create a wave file in which we put the header of the original wave file followed by the new audio data.

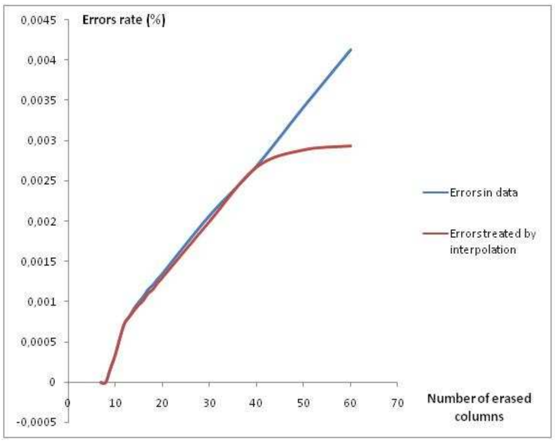
5. RESULTS

All the following tests have been made with audio.wav.

5.1 Single scratch on our data

Our first experimentation consists in simulating a single scratch on a CD. To do this, we erase a block (successive columns) of data and the number of erased columns stands for the length of the scratch. The operation is made by the function “Rayure1” that erased a certain number of columns in a specific sector. We observe the rates of residual errors and of errors corrected by interpolation on the following graphic:

Rates evolutions of residual errors (in blue) and of errors treated by interpolation (in red) according to the number of columns erased by a single scratch



This graphic shows that the interpolation can recover until 42 successive erased columns. Yet, theoretically, the interpolation is supposed to treat 48 successive columns. Our algorithm of interpolation is clearly not optimal but it has an efficiency of 87, 5%.

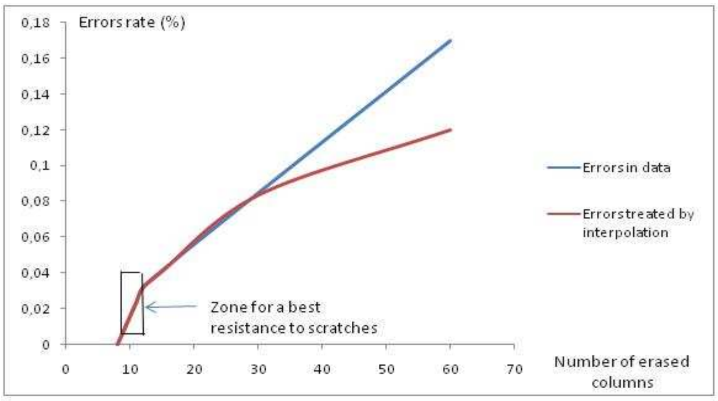
Moreover, our CIRC decoding can correct 8 successive wrong frames, which corresponds to an efficiency of 53%. We can explain that because we do not implement the most optimal CIRC decoding.

5.2 Quality of interpolation after several scratches

In this test, we make scratches of C columns in one sector out of two for 70 sectors. The first modified sector is the 1000th one. So, we observe the phenomenon (we hear strong clicks that can be harmful to the headphones) during one second at the 13th second of the audio file.

Test	Number of modified codes	Frequency of columns modification	Frequency of sectors modification	Modified bytes of codes	Number of errors in data	Number of errors corrected after interpolation	Quality Of data protection
testmauvais1 / testmauvais-inter1	3	1/4	1/3	1101	6817	6827	Very bad
Testmauvai2 / testmauvais-inter2	3	1/4	1/7	1101	2994	2964	Very bad
testmauvais3/ testmauvais-inter3	3	1/4	1/10	1101	2098	2086	Bad
testmauvais4 / testmauvais-inter4	3	1/4	1/15	1101	1369	1355	Bad
testqualite2 / testqualite-inter2	3	1/4	1/18	1101	1197	1183	Ok
testqualite4 / testqualite-inter4	3	1/4	1/20	1101	1067	1061	Ok
testqualite5 / testqualite-inter5	3	1/4	1/30	1101	716	712	Good
testqualite6 / testqualite-inter6	3	1/4	1/40	1101	541	539	Good
testqualite7 / testqualite-inter7	3	1/4	1/50	1101	431	429	Good
testqualite8 / testqualite-inter8	3	1/4	1/100	1101	208	208	Very Good

Evolution of the errors rate according to the number of columns erased by a scratch on one sector out of two and for 70 sectors

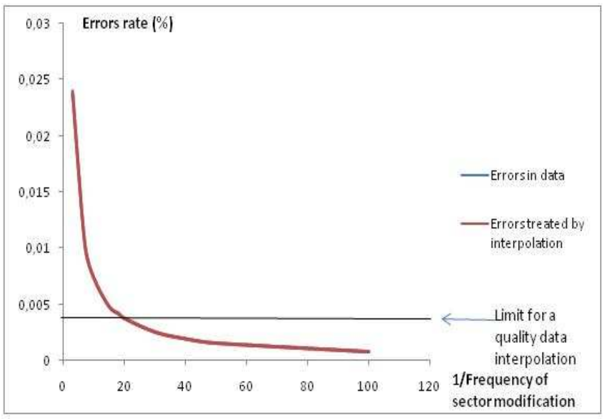


The data read with interpolation are not satisfying: we still hear an unpleasant sound. So, our implementation of interpolation is limited in this particular case. We cannot plan to protect our data this way.

5.3 Quality of interpolation after inserting errors on Reed-Solomon codes

In this last experimentation, we make a series of tests with an insertion of errors in Reed-Solomon codes in order to modify the decoded data. We set the number of modified columns to 3 and the frequency of columns modification to ¼ and we try to obtain the best analogical reading with a disturbing quality of sound (too many non periodic clicks) during the numerical reading.

Evolution of the errors rate according to the frequency of sector modification with a number of erased columns (per sector) of 3 and a frequency of columns modification of 1/4



We fix the optimal parameters of our method to a number of modified bytes in the C1 and C2 codeword of 3, a frequency of sector modification of 1/100 and a frequency of columns modification of ¼. The interpolation obtained with these parameters is optimal: all the errors are treated and the sound is as clean as the original audio file. So, the audio protection is sure.

6. CONCLUSION

Our goal has been reached because we have implemented a working solution that uses the technique of errors insertion. However, we would like to have more time to try to implement the other methods mentioned in the present report (the modification of the TOC particularly) so as to mix them and to realize a complete CD protection. Besides, we could also improve the method we implemented (CIRC decoding, temporal interpolation) in order to have better results.

This project was interesting because the theme was a current subject with concrete hardships like those we will have to face in an incoming future.

REFERENCES

[1] Georges Zénatti, *CD-Rom et vidéo sur CD*, Hermès, 1996.

[2] Heitaro Nakajima and Hiroshi Ogawa, Compact Disc Technology, Ohmsha, 1992.

[3] Joan Feigenbaum , Digital Rights Management, Springer.

[4] J. Alex Halderman and Edward W. Felten, Lessons from the Sony CD DRM Episode, Department of Couputer Science, Princeton Univeristy, 2006.

[5] Jean-Pierre Zanotti, Codage d’un signal audio-numérique, 1998.