A Coding Approach for Detection of Tampering in Write-Once Optical Disks

Mario Blaum Jehoshua Bruck* Kurt Rubin Wilfried Lenth

IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120
{blaum,rubinka,lenth}@almaden.ibm.com

*California Institute of Technology
Mail Stop 136-93
Pasadena, CA 91125
bruck@paradise.caltech.edu

Abstract

We present coding methods for protecting against tampering of write-once optical disks which turns them into a secure digital medium for applications where critical information must be stored in a way that prevents or allows detection of an attempt at falsification. Our method involves adding a small amount of redundancy to a modulated sector of data. This extra redundancy is not used for normal operation, but can be used for determining, say as a testimony in court, that a disk has not been tampered with.

Keywords: Optical Disks, (d, k) codes, Write-Once Memory, Data Integrity, Tampering.

*Research was supported in part by the NSF Young Investigator Award CCR-9457811 and by the Sloan Research Fellowship.

1 Introduction

In documents to be presented in court, it is essential that the integrity of the document can be guaranteed without a doubt. To this end, techniques like digital signatures [11] [14] are used. These techniques normally involve cumbersome numerical operations, that create unacceptable delays when the write operation is implemented. Moreover, the reliability of these techniques is under continuous controversy. The purpose of this paper is to present an efficient method for detecting possible tampering of optical disks while requiring a relatively small addition of redundant information. The method also tolerates a number of random errors, thus eliminating situations of false alarms.

Optical disks contain several layers of protection that ensure the integrity of the data. The data is protected by ECC and CRC codes [6] for correction and detection of errors, respectively. The attacker may attempt to rewrite this data. However, if a write-once recording media is used, the attacker cannot erase any written data. This is because write-once optical media use recording mechanisms that leave an irreversible materials change. They are therefore a good starting point for secure digital media. What is needed are additional coding approaches that prevent information tampering via formation of additional marks or adding to the length of already written marks. The particular physics of the write process, in which the laser marks correspond to 1's that cannot be erased, provide an opportunity to create a very simple digital signature, that will guarantee with total certitude that the optical disk has not been altered.

This problem was first studied in [9]. There, the author proposes the use of unordered codes (like for instance, constant weight codes), to detect hostile tampering. There are two problems with this approach that seriously undermine its value in practical applications: the first problem is that it assumes unconstrained data at the encoding, ignoring the particular modulation codes used in optical recording, like $(1,7)$ or $(2,7)$ codes [8]. The second problem is that the normal noise existing in the system, like scratches, corrosion, soft errors, etc., cannot be distinguished from hostile tampering of the data. We will present a realistic method that overcomes these two problems. In particular, our method is based on the concept of EC/AUED (Error Correcting/All Unidirectional Error Detecting) codes [4][5] and has the following features:

1. It can be used with modulated data. In fact, the preferred embodiment that we describe is for data modulated using a $(1,7)$ code.

2. It can be tuned to operate in the presence of a prescribed number of errors. Namely, false alarms caused by normal noise can be eliminated.

3. The representation of the data is not altered by our method. Only a small amount of redundant information is added to secure the data.

4. The method does not affect the read operation. The redundant information is accessed only while performing the verification procedure.

5. The encoding and verification procedures are easy to implement.

The paper is organized as follows: the next section describes modulation techniques used in write-once optical disks and a simple method that takes into account the modulation codes but might generate false alarms due to noise in the reading process. In Section 3, we present a general method that can tolerate any number of errors appearing as normal noise, and at the same time, can detect hostile tampering.

2 Tampering in a noiseless environment

A key aspect of providing security by unambiguous detection of tampering is to use a recording technology in which information is written in a permanent manner. Data storage media which are erasable, such as magnetic storage or erasable optical storage, either phase change or magneto-optic, are not suitable for this security application since information could be tampered with or changed without leaving evidence that changes had been made. Fortunately, there are non-erasable versions of data storage media which readily lend themselves to implementing a scheme to detect tampering. A common type is write-once optical storage media, which comes in the form of disks, cards and tape [13]. These are media in which data can only be written once and cannot be erased once it is written.

Write-once optical storage media typically consists of a light absorbing thin film recording layer deposited on a substrate. Marks are formed at specific spatial locations of the recording layer, as dictated by the data encoding pattern, when a sufficiently intense focused laser beam is incident upon the recording layer. Heating of this layer causes an irreversible transformation to occur locally in the region of the focused beam. These permanent transformations include but are not limited to:

1. amorphous to crystalline phase change [10],
2. chemical reactions such as decomposition of dyes [15] which are used in CDR media, and
3. forming permanent holes by ablation [7].

Each of these transformations causes a localized change of reflectivity which is detected to indicate the presence of a mark. Since the transformations are irreversible, there is no way to undo or erase the written data. However, it would be possible to go back to the previously written mark and make it longer or make a new mark in between two previously written marks.

Marks can be recorded on an optical disk in a number of ways. The two most common are pulse position modulation (PPM) and pulse width modulation (PWM). PPM recorded data uses marks of only one type which are essentially circular in shape. Information is encoded by the presence or absence of the mark. In the case of (2,7) [8] encoded data, the readback signal from the circular mark would be approximately 3 channel clocks long. Each written mark would be separated from the next circular mark by a space which could be from 3 to 8 channel clocks long. PWM recording actually encodes information on both the leading and trailing edges of the written mark which yields a substantially higher linear density than PPM for the same minimum mark size. In the case of (1,7) encoded PWM data, the minimum mark size would be 2 channel clocks long. This mark would be essentially circular in shape. Longer marks can also be formed, up to a maximum of 8 channel clocks for this code. Following each mark, there is a space. The minimum space is 2 channel clocks long and the maximum space is 8 channel clocks long. It is in the spaces between the marks, regardless of whether it is PPM or PWM encoded data, an attacker could potentially add another mark which would change the encoded data.

In the rest of the paper, we will concentrate essentially on the PPM writing and then explain how to extend the results to PWM. We will assume that each mark is a 1 while the absence of a mark is a 0. In general, the most common modulation codes are the so called (d, k) constrained codes [8]. Namely, each pair of consecutive 1's is separated by at least d 0's and by at most k 0's. The attacker cannot erase the marks (i.e., the 1's), but can make marks as he pleases on the 0's. For instance, the compact disk uses a (2,10) modulation code [8] and future products envision the use of a (1,7) modulation code [1]. Although in this paper we will focus on (d, k) modulation codes, our techniques can be easily extended to address other modulation schemes.

We propose adding an extra string (that we call a *tail*) at the end of the data sector such that the writing of extra marks will be detected. Note that the tail does not need to be physically located at the end of the sector, but can be anywhere in the disk. The idea in this section is to create an unordered code similarly to the Berger construction [3]. The difference between our construction and the Berger construction is that we need to take into account the (d, k) constraints.

Let us describe the method before giving an example and formalizing it. Assume that we encode M bits of data into N bits using a (d, k) constrained code. Let n be the number of 1's in this modulated string. Notice that the minimum value of n occurs only when we have runs of k 0's followed by a 1, and there can only be $N/(k+1)$ such runs. Explicitly, this string has the form

$$\overbrace{\overbrace{00\ldots0}^{k}1\ \overbrace{00\ldots0}^{k}1\ \ldots\ \overbrace{00\ldots0}^{k}1}^{N}\,.$$

On the other hand, the maximum value of n occurs when we have have runs of d 0's followed by a 1, and there can only be $N/(d+1)$ runs. Explicitly, this string has the form

$$\overbrace{\overbrace{00\ldots0}^{d}1\ \overbrace{00\ldots0}^{d}1\ \ldots\ \overbrace{00\ldots0}^{d}1}^{N}\,.$$

Therefore, we have to represent $(N/(d+1))-(N/(k+1))+1$ possible numbers while following the roles of the modulation.

The following is a systematic encoder for a rate 2/4 modulation code:

$$\begin{aligned}
00 &\leftrightarrow 0010\\
01 &\leftrightarrow 0001\\
10 &\leftrightarrow 0100\\
11 &\leftrightarrow 0101
\end{aligned}$$

We call this code $S(2,4)$. Note that the code is systematic, since the second and the fourth coordinates represent the information bits.

For example, the string 00101011 will be encoded using $S(2,4)$ to the string 0010010001000101. Observe that the code $S(2,4)$ is actually a $(1,5)$ modulation code. We note here that $S(2,4)$ can be made a (1,3) modulation code by varying the first bit between 0 and 1. Next we present the encoding algorithm.

Algorithm 2.1 (Modulated-Berger Encoding Algorithm)
Let n be the number of 1's in the modulated sector. Then the tail (the redundant bits) is the complement of the number $n-(N/(k+1))$ modulated using the code $S(2,4)$. Note that, in case the data and the tail are adjacent on the disk, we might need to add at most d merging bits to connect between the data and the tail while complying with the (d,k) constraints.

Example 2.1 Assume that we want to encode (1,7)-constrained sequences of length 32. The minimum number of 1's is 4, which occurs when the sequence

00000001000000010000000100000001

is received, while the maximum number of 1's is 16, which occurs when the sequence

01010101010101010101010101010101

is received. Therefore, we need to represent $16-4+1=13$ different values. Namely using $S(2,4)$ we need 8 bits.

Assume that the following (1,7) sequence of length 32 has been written:

$$0010100010101000000100010010001 0.$$

Counting the number of 1's we obtain $n = 9$. Then, we have to write the complement of $9 - 4 = 5$. The number 5 is 0101 and its complement is 1010.

The encoding then gives the following sequence

$$00101000101010000001000100100010\ 0100\ 0100$$

Now assume that an attacker tries to tamper the data and writes 1's in places that were 0's. Moreover, the receiver gets the following sequence:

$$00101010101010101001010100101010\ 0100\ 0100$$

When the disk is examined, we count 14 1's. That corresponds to $14 - 4 = 10$ which is 1010. The complement is 0101 and the encoding of the tail is: 0001 0001. We see that it differs from 0100 0100 which is the tail stored in the disk, so tampering is suspected. There is no way that the attacker may tamper the data in such a way that tail 0100 0100 becomes tail 0001 0001 (he would need to erase 1's to do so).

The tail does not need to be appended to the end of the sector, but can be written in a dedicated part of the disk. In normal read operation, the security tail is ignored. However, when checking data from the disk for possible tampering, all we have to do is recompute the tail using Algorithm 2.1 and compare it with the tail already stored in the disk: if the two coincide, then no tampering is declared. Otherwise, tampering is suspected. The next theorem states that this is the case.

Theorem 2.1 Assume that an attacker tampers data written using Algorithm 2.1. Then, the tampering will be detected, i.e., the tail computed by using the tampered data will not coincide with the tail written in the disk.

Proof: Our construction is combining between the Berger construction of systematic unordered codes [3] and the modulation using $S(2, 4)$. Notice that the attacker can only increase the number n of 1's in the data and in the tail when he tampers the information. Assume that the attacker increases the number of 1's in the information part, then he should be able to erase marks in the tail in order to create an appropriate tail, which is impossible in the context of this problem. $\square$

Next we consider an example more related to actual applications.

Example 2.2 Consider a 1024-byte sector, each byte consisting of 8 bits. This gives a total of 8192 bits. When modulated into a rate 2/3 (1,7) code, we obtain a total of $(3/2)8192 = 12288$ bits. The next step is counting the number n of 1's in this modulated string. Notice that the minimum value of n occurs only when we have runs of 7 0's followed by a 1, and there can only be $12288/8 = 1536$ runs. Explicitly, this string has the form

$$\overbrace{00000001\ 00000001\ \ldots\ 00000001}^{12288}.$$

On the other hand, the maximum value of n occurs when we have the run 01 repeated $12288/2 = 6144$ times. Explicitly, this string is

$$\overbrace{01\ 01\ \ldots\ 01}^{12288}.$$

Therefore, we have to represent $6144 - 1536 + 1 = 4609$ possible numbers.

In order to represent 4609 numbers, we need 13 bits, namely, using Algorithm 2.1 sequence of length 26 bits. If we add a merging bit, the total length of the tail is 27 bits.

The approach presented above is not the most efficient in terms of the amount of redundancy. However, it is very easy to implement. Next we present an optimized way of representing the tail using a look-up-table encoding of the bit strings to (d,k) modulated strings.

In order to represent these $(N/(d+1)) - (N/(k+1)) + 1$ numbers, we follow the method of Beenker and Immink [2]. The method of Beenker and Immink consists of considering $(N/(d+1)) - (N/(k+1)) + 1$ (d,k) sequences of length m starting with at most $k-1$ 0's and ending with at most $k-1$ 0's. Preferably, we will choose m minimal with these properties. In addition, d merging bits that avoid violation of the (d,k) constraints are added at the beginning of the string, giving a total of $m+d$ bits for the tail.

Next we order these sequences in non-increasing weight order. Let us call T the $((N/(d+1)) - (N/(k+1)) + 1) \times m$ matrix whose rows are the sequences described above, and denote by $\underline{t}_i$ the i-th row of T, $1 \le i \le (N/(d+1)) - (N/(k+1)) + 1$. Therefore, if $i < j$, then $w_H(\underline{t}_i) \ge w_H(\underline{t}_j)$, where $w_H(\underline{t}_i)$ denotes the Hamming weight of $\underline{t}_i$. Having defined the matrix T, we add an $(m+d)$-bit tail to the sector by using the following algorithm:

Algorithm 2.2 (Look-Up-Table Encoding Algorithm)
Let n be the number of 1's in the modulated sector. Then, add d merging bits followed by the vector of length m $\underline{t}_{n-(N/(k+1))+1}$. The merging bits are obtained according to the method in [2].

Notice that for an increasing value of n, the number of 1's in the tail as given by Algorithm 2.2 is non-increasing.

Example 2.3 Assume that we want to encode (1,7)-constrained sequences of length 32. The minimum number of 1's is 4, which occurs when the sequence

$$00000001000000010000000100000001$$

is received, while the maximum number of 1's is 16, which occurs when the sequence

$$01010101010101010101010101010101$$

is received. Therefore, we need 16-4+1=13 (1,7) sequences. For instance, the following 13×6 matrix provides modulated sequences with weights in non-increasing order:

$$B = \begin{pmatrix} 000101 \\ 001001 \\ 001010 \\ 010001 \\ 010010 \\ 010100 \\ 010101 \\ 100001 \\ 100010 \\ 100100 \\ 101000 \\ 000001 \\ 000010 \end{pmatrix}$$

Assume that the following (1,7) sequence of length 32 has been written:

$$001010001010100000001000100100010.$$

Counting the number of 1's we obtain $n = 9$. Then, we have to append the merging bit followed by row 9-3=6 of matrix B, which is

$$010100.$$

The encoding then gives the following sequence

$$001010001010100000001000100100010\;1\;010100.$$

Now assume that an attacker tries to tamper the data and writes 1's in places that were 0's. Moreover, the receiver gets the following sequence:

$$00101010101010101001010100101010\,1\,010100.$$

When the disk is examined, we count 14 1's. That corresponds to row 14-3=11 of B, which is

$$101000.$$

We see that it differs from 010100, which is the tail stored in the disk, so tampering is suspected. There is no way that the attacker may tamper the data in such a way that tail 010100 becomes tail 101000 (he would need to erase 1's to do so).

Consider as in Example 2.2 the 1024-byte sector, each byte consisting of 8 bits. The method of Beenker and Immink in this case consists of considering all the (1,7) sequences of length 18 starting with at most 6 0's and ending with at most 6 0's. In addition, a merging bit that avoids violation of the (1,7) constraints is added at the beginning of the string, giving a total of 19 bits. There are more than 4609 sequences using the method described above. We can order these sequences in non-increasing number of 1's as follows: consider first 9 sequences of weight 7 in alphabetical order, then all the sequences of weight 6 in alphabetical order (there are exactly 1709 of them), then the 1876 sequences of weight 5, and finally the 1015 sequences of weight 4, giving a total of 4609 sequences. Let T be the 4609×18 matrix whose rows are the sequences described above, and let $\underline{t}_i$ be the i-th row of T, $1 \leq i \leq 4609$. Then, we add a 19-bit tail to the sector by using Algorithm 2.1. In this particular case, the merging bit is 1 if the preceding and the following bits are 0, and it is 0 otherwise.

In summary, we have a trade-off between the two methods. The first method provides efficient encoding/decoding. However, it produces longer tails. The second method requires large look-up-tables for encoding/decoding but produces shorter tails.

3 Tampering in a noisy environment

Errors like natural scratches, corrosion, soft errors, etc., may create a mismatch between the read sector and the tail in the process described in Section 2. If the disks are stored during a long period of time, then it is very likely that in many cases the evidence will be dismissed as unreliable when the underlying error-correcting code can perfectly take care of the noise. Thus, it is desirable to device a method that allows to distinguish between natural noise and a real possibility of data tampering.

We next give a new algorithm that allows for a certain number of errors, say t_1, in the information part, and t_2 errors in the tail. As in Section 2, we assume that there is a systematic encoder that encodes a vector into a (d,k)-constrained code. For instance, for $d = 1$, we have $S(2,4)$ which is a (1,5) code of rate 2/4. Normally, the rate of a systematic

(d,k)-constrained code is worse than the rate of a (d,k) code without the restriction that it must be systematic. However, this is not critical since we are going to encode into a systematic (d,k) code only a very short vector.

There is another concept that we need to introduce in order to describe the Encoding Algorithm. It is the concept of a t-error-correcting/all unidirectional error-detecting (EC/AUED) code [4][5]. In order to define a t-EC/AUED code, we need some notation. Let $\underline{u}$ and $\underline{v}$ be binary vectors of the same length. Then, $N(\underline{u},\underline{v})$ denotes the number of coordinates where $\underline{u}$ is 1 and $\underline{v}$ is 0. For example, $N(1101,0110)=2$ and $N(0110,1101)=1$. Notice that $N(\underline{u},\underline{v})+N(\underline{v},\underline{u})=d_H(\underline{u},\underline{v})$, where d_H denotes Hamming distance.

We say that a binary code $\mathcal{C}$ is t-EC/AUED, if, given any ordered pair $\underline{u}$ and $\underline{v}\in\mathcal{C}$, then $N(\underline{u},\underline{v})\geq t+1$. For example, let

$$\mathcal{C} \;=\; \{001110\;,\;110010\;,\;011001\}.$$

We can easily verify that $N(\underline{u},\underline{v})\geq 2$ for any $\underline{u},\underline{v}\in\mathcal{C}$, therefore, $\mathcal{C}$ is 1-EC/AUED. Describing efficient constructions as well as encoding and decoding algorithms of t-EC/AUED codes is beyond the scope of this paper, so again we refer the reader to [4][5].

We say that a binary vector has suffered unidirectional errors, if all the errors in the vector are either of type $0\rightarrow 1$ or $1\rightarrow 0$. The main property of a t-EC/AUED code is the following: it can correct up to t random errors and detect any number of unidirectional errors when this number exceeds t. Next we will show how to use this property of t-EC/AUED codes to detect tampering.

Let t_1 be the number of error that we are willing to tolerate in the information part and t_2 the number of errors in the tail. Normally, t_1 is much larger than t_2 and is related to the number of errors that the underlying ECC can handle. In other words, t_1 is related to the normal noise of the system, and we want to avoid that this noise will make us suspect that tampering has occurred, as would be the case if we used the method described in the previous section. Next, we describe an encoding method to produce a tail that is resistant to normal noise. Interestingly, t_1 will have no role in the encoding, and can be determined, varied, or made adaptive, according to the system. We use the notation of the previous section where relevant.

Algorithm 3.1 (EC/AUED Encoding Algorithm)
Let n be the number of 1's in an N-bit (d,k)-constrained sector. Then:

1. Let $\underline{u}$ be the $\lceil \log_2((N/(d+1))-(N/(k+1))+1)\rceil$-bit vector representing n in binary.

2. Let $\underline{v}$ be the encoding of $\underline{u}$ into a t_2-EC/AUED code [4].

3. Let $\underline{t}$ be a systematic encoding of $\underline{v}$ into a (d, k)-constrained code.

Then, $\underline{t}$ is the tail vector appended to the sector (possibly with d merging bits).

The parameter t_1 a priori determined is utilized during the verification algorithm to be presented next.

Algorithm 3.2 (Verification Algorithm) Assume that $(\underline{s}, \underline{t})$ is received, where $\underline{s}$ represents the sector and $\underline{t}$ represents the tail. Let n be the number of 1's in $\underline{s}$. Then:

1. Demodulate $\underline{t}$ to obtain the vector $\underline{v}$.
2. Decode $\underline{v}$ for t_2 errors. If no decoding is obtained, then tampering is suspected. Otherwise, let $\underline{u}$ be the information bits corresponding to the decoding of $\underline{v}$.
3. Let m be the number represented by the binary vector $\underline{u}$. If $|n - m| \leq t_1$, then accept the sector. Otherwise, tampering is suspected.

The next theorem proves that the method is 100% efficient against tampering.

Theorem 3.1 Assume that an attacker tampers data written using Algorithm 3.1, creating either $> t_1$ marks in the information part or $> t_2$ marks in the tail. Then, this tampering will be detected by Algorithm 3.1.

Proof: Assume first that the attacker made $> t_2$ marks in the tail, and without loss of generality assume that those marks affect the systematic part of the tail only. In particular, since he can only make 1's, this is equivalent to creating $> t_2$ unidirectional errors, which will be detected by the code and tampering is suspected.

Assume now that the attacker makes $> t_1$ marks in the information part and $\leq t_2$ marks in the tail. Then, the number m represented by $\underline{u}$ is correctly obtained. However, n, the number of 1's in $\underline{s}$, is greater than m and $n - m > t_1$, thus, tampering is suspected. □

If the attacker makes $\leq t_2$ marks in the tail, these marks will be corrected by the t_2-EC/AUED code. If he makes $\leq t_1$ marks in the information part, we are detecting this fact, but we are trusting the correction of errors to the ECC, which we a priori know.

The next example will illustrate Algorithms 3.1 and 3.2.

Example 3.1 Consider the situation of Example 2.1. Assume that we fix $t_1 = 2$ and $t_2 = 1$. To write the weights, we needed 13 numbers, therefore, 4 bits. These 4 bits can be encoded into at 1-EC/AUED code of length 11 as described in [4].

As in Example 2.1, assume that the following (1,7) sequence of length 32 has been written:

$$00101000101010000001000100100010.$$

Next we obtain the tail $\underline{t}$ according to Encoding Algorithm 3.1.

The number of 1's is $n = 9$. The binary representation of 9 is $\underline{u} = 1001$. Using the $[7, 4, 3]$ Hamming code with parity-check matrix

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$\underline{u}$ is encoded as $\underline{u}' = 1001001$. Using the descending tail matrix [4]

$$\begin{matrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{matrix}$$

vector $\underline{u}'$, which has weight 3, is encoded as $\underline{v} = 10010011010$. Finally, we modulate $\underline{v}$ using the (1,5) rate 1/2 code. Since $\underline{v}$ is 11 bits long, we modulate the first bit as

$$0 \leftrightarrow 00$$
$$1 \leftrightarrow 01$$

The remaining 10 bits are modulated in pairs in the usual way. Therefore, $\underline{v}$ is modulated as

$$\underline{t} = 0100100100000101000100.$$

Finally, by adding a merging bit as described in the previous section, the final encoded information with tail is then

$$00101000101010000001000100100010 \ 1 \ 0100100100000101000100.$$

As before, assume that an attacker tries to tamper the data by writing 1's in places that were 0's. We will now use Verification Algorithm 3.2. Assume that the receiver gets the following sequence:

$$0010100010101001000101010010001 0\ 1\ 0100100101000101000100.$$

The first step is demodulating the tail. Doing so, we obtain the vector 10011011010. Consider the first 7 bits corresponding to the [7,4] Hamming code, i.e., 1001101. The decoding of this vector gives 1001001 (one error in the 5th bit). Recomputing the last 4 bits, we obtain 1010, which corresponds to the received vector. Therefore, the tail is accepted, and the information part is 1001, which corresponds to the number 9. Now, counting the number of 1's in the information part, we see that this number is 11. Since 11-9=2 and $t_1 = 2$, we accept the information.

On the other hand, assume that the receiver gets

$$00101000101010010101010100101010\ 1\ 0100100101000101000100.$$

In this case, the tail is the same as before, so it is accepted. The information part has now 13 1's, and $13 - 9 = 4 > 2$, so tampering is suspected.

Now, let us consider a more realistic situation as at the end of Section 2. Consider a 1024-byte sector, each byte consisting of 8 bits. Thus, we need a 13-bit binary vector $\underline{u}$ to represent the number of 1's in a (1,7)-constrained sector. Assume that we fix $t_2 = 4$. In Algorithm 3.1, vector $\underline{u}$ has to be encoded into a 4-EC/AUED code. Following the methods in [5] or [4], $\underline{u}$ can be encoded into a 4-EC/AUED code of length 38. Using the (1,5) systematic modulation code of rate 1/2, we obtain a tail $\underline{t}$ of length 76. We can also append a merging bit between the tail and the sector to prevent a violation of the 7-constraint. The parameter t_1 may be arbitrarily determined according to the specifications of the product.

The method can be adapted to PWM type of writing with little modifications. In a PWM setup, marks have different lengths, corresponding to strings of 1's. Now a tamperer has two possible strategies: one is increasing the length of a mark (i.e., to a string of 1's), and another one is writing new marks between old marks. In either case, the total number of 1's increases by this action of the tamperer. This allows us to use Berger codes to write down the number of 0's (i.e., the total space between the marks). The tamperer can only decrease the number of 0's, allowing for easy detection using the Berger method. The method can be adapted for different runlengths of symbols.

4 Conclusions

We have presented a method that can be used to detect possible tampering of Write-Once storage media. The method concentrates on preventing the rewriting of a disk. It is very general and provides total protection against tampering.

The paper has focused on implementing the security scheme in write-once optical disks. However, the concept is readily generalized to other 2-dimensional, planar-based write-once recording technologies such as optical tape or AFM probe storage. In addition, the concept can be generalized to 3-dimensional write-once storage media such as multilayer [12] or holographic storage. In this case, a sector might be 3-dimensional in nature but the same methodology of counting the number of recorded user bits and encoding that number would still apply. In addition this concept is readily extendible to any other type of permanent recording technology, such as programmable read only memories (PROM) which may be implemented in smart cards.

References

[1] R. Adler, M. Hassner and J. Moussouris, "Method and Apparatus for Generating a Noiseless Sliding Block Code for a (1,7) Channel with Rate 2/3," U.S. Patent 4,413,251, 1982.

[2] G. F. M. Beenker and K. A. S. Immink, "A Generalized Method for Encoding and Decoding Run-Length-Limited Binary Sequences," IEEE Trans. Inform. Theory, vol. IT-29, no. 5, pp. 751–754, Sept. 1983.

[3] J. M. Berger, "A Note on Error Detecting Codes for Asymmetric Channels," Information and Control, vol. 4, pp. 68-73, 1961.

[4] M. Blaum and H. van Tilborg, "On t-Error Correcting/All Unidirectional Error Detecting Codes," IEEE Trans. on Computers, pp. 1493–1501, Nov. 1989.

[5] J. Bruck and M. Blaum, "New Techniques for Constructing EC/AUED Codes", IEEE Transactions on Computers, Vol. 41, No. 10, pp. 1318–1324, October 1992.

[6] N. Glover and T. Dudley, "Practical Error Correction Design for Engineers," Data Systems Technology, Corp., 1988.

[7] M. Horie, T. Tamura, M. Ohgaki, H. Yoshida, Y. Kisaka, Y. J. Kobayashi, "TeSeF films by TeSe SeF6-Ar Reactive Sputtering for Ablative Optical Recording," J. Appl. Phys. Vol.75, No. 5, pp. 2680–2689, March 1994.

[8] K. A. S. Immink, "Coding Techniques for Digital Recorders," Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[9] E. L. Leiss, “Data Integrity in Optical Disks,” IEEE Trans. on Computers, pp. 818–827, Sept. 1984.

[10] T. Nishida, H .Sugiyama, S. Horigome, “SnSbSe/SbBi Bilayer Phase Change Media for High Density Write Once Optical Recording,” Jpn. J. Appl. Phys. 1, Regul. Pap. Short Notes (Japan), Vol.34, No.3, pp. 1562–1568, March 1995.

[11] F. Piper and N. Stephens, “Digital Signatures: RSA or El Gamal?,” Cryptography and Coding III, edited by M. J. Ganley, Oxford University Press Inc., New York, pp. 311–320, 1993.

[12] K. A. Rubin, H. J. Rosen, W. W. Tang, W. Imaino, T. C. Strand, “Multilevel Volumetric Optical Storage,” Proc. SPIE - Int. Soc. Opt. Eng., Vol. 2338, pp. 247–253, 1994.

[13] A. J. G. Strandjord, S. P. Webb,D. J. Perettie, R. A. Cipriano, “Flexible Storage Medium for Write Once Optical Tape,” Proc. SPIE - Int. Soc. Opt. Eng., Vol. 1663, pp. 362–371, 1992.

[14] H. C. A. van Tilborg, “An Introduction to Cryptology,” Kluwer Academic Publishers, 1988.

[15] C. Yokota, T. Sasakawa, H. Hyakutake, “Phthalocyanine CDR for High Speed Recording,” Proc. SPIE - Int. Soc. Opt. Eng. (USA), Vol. 2514, pp. 249–257, 1995.