AN IMPROVEMENT ON CODE FOR DETECTION OF TAMPERING IN WRITE-ONCE OPTICAL DISKS

Ching-Nung Yang

Department of Computer Science & Information Engineering,
National Dong Hwa University,
1, Sec. 2, Da Hsueh Rd., Shou-Feng, Hualien, Taiwan, Republic of China
TEL: (03)8662500 Ext-22120 FAX: (03)8662781
E-mail: cnyang@mail.ndhu.edu.tw

ABSTRACT

It is obvious that we can protect the integrity of the data in write-once optical using cryptographic techniques like digital signatures. However, a code for detection of tampering in write-once optical disks without using digital signature, is proposed in [1]. The technique in [1] is to choose a t-error correcting/all unidirectional error detecting (t-EC/AUED) code as a security tail, such that we can check possible tampering. The tail for a sector of data is a function of the weight of the sector. Here, We present a new look-up-table method to construct the security tail by using constant weight t-EC/AUED codes. It is shown that our method has the smaller tail than the previously developed coding approach in [1]. This paper also defines a new class of code with tampering detecting capability, and the code of tampering detection in [1] is only a special case in the new class of code.

1 INTRODUCTION

Digital signatures can be used to guarantee data integrity, but when using digital signature to protect the data, the two followings cannot be guaranteed. Here, we quote a statement from the introduction of [1] below : *It can not be assured that an attacker having a great mount of time(like several years) cannot break the system. Moreover, the decryption involves knowledge of a key. It is conceivable that an attacker may gain physical possession of the key, in which case the cryptographic techniques are useless.* Thus, we use coding method instead of digital signature to protect the data in write-once optical disks.

There are two important properties in write-once optical disks. The first property is that the error pattern of write-once optical disks is asymmetric, and the second property is that the particular codes runlength-limited (RLL) (d, k) codes are used in optical disks.

The error that changes a bit in binary word from 0(1) to 1(0) is denoted as 0-error(1-error). Both 0-error and 1-error can occur but they do not occur in the same codeword, called as unidirectional errors. There is only one type of errors, say 0-error; and the other type of errors, say 1-error, will never occur in any codeword, called asymmetric errors. Many papers on dealing with the codes for detecting and correcting asymmetric/unidirectional errors [6]-[13] have been published. It is observed that the error pattern of write-once optical disks is asymmetric due to the irreversible materials usage when recording. So, there is no way to erase the data from "1" to "0", but it is possible to make a new "1" in all-0 areas.

The most frequently modulation code applied in write-once optical disks is RLL (d, k) sequence [5]. It means that two logical "ones" are separated, at least, d consecutive "zeros", at most k "zeros". The parameter d is to avoid the intersymbol interference (ISI), and the parameter k is to maintain the bit synchronization of system clock. It is obvious that a (d, k)-constrained code of length n has the weight span from $n/(d+1)$ to $n/(k+1)$.

How to achieve the purpose of data integrity in write-once optical disk, the first method proposed in [2] is to use unordered code. However, it does not consider the modulated (d, k)-constrained codes. In [1], it combines (d, k) codes and t-EC/AUED codes, and get a good result. There are two coding approaches in [1]. One is the detection of tampering in a noiseless environment and the other is in a noisy environment, where "noisy" means that errors caused by natural scratches, corrosion, .., etc., not tampering by external attacker. We here only give the improvement for coding in a noisy environment.

The coding approach for detection of tampering in a noisy environment [1] :

The encoding procedure of this scheme is shown in Figure 1. The code can be tolerate in the information part at most t_1 normal errors and detect all possible tampering errors, and correct t_2 errors in the tail (abbreviated as (t_1, t_2)-code). In general, t_1 is larger than t_2.

We use the following example to describe the operation process.

Example 1 : Assume we want to encode a 16 bits original data $\underline{m}$= (0000101011111000) with t_1=2, t_2=1. Here, we use two RLL codes (1, 7) code with code rate=2/3 and systematic (1, 5) code with code rate=1/2 which can be found in [1], [3], [4], [5], and a 1-EC/AUED code in [9]. First use (1, 7) RLL code encodes original data $\underline{m}$ into information part $\underline{s}$=(101000001001010010001000) of length 24=16×3/2. Since $W(\underline{s})$=7, which $W(\underline{s})$ is Hamming weight of $\underline{s}$, and its weight distribution between 3(=24/8) and 12(=24/2),i.e., ten different weights. So, we can use 4-tuple $\underline{u}$ (since $2^4 \geq 10$) to represent "7" as "0111". Using 1-EC/AUED, we encode $\underline{u}$ to get 11 bits output $\underline{v}$=(01110011010). Due to the criterion of (1, 7) constraint, we use systematic (1, 5) RLL code to get the 22 bits tail $\underline{t}$=(0001010100000101000100).

In this paper, a look-up-table method is proposed to reduce the length of tail, and the table size is reasonable. We use non-systematic constant weight t-EC/AUED code [6], [12], [13] as the tail part. As we know, the encoding/decoding algorithm of group theoretical t-EC/AUED code needs many recursive computations [12], [13]. However, now it does not need the encoding/decoding of a non-systematic t-EC/AUED code, but only needs a table to set the mapping between a weight and a non-systematic t-EC/AUED codeword.

In this paper, we also discuss a more general approach for tampering in a noisy environment in [1], a (t_1, t_2, t_3)-code, where t_3 is called as tampering detecting capability. A (t_1, t_2, t_3)-code will result a smaller tail due to the compromise of tampering detecting capability. It is obvious that a (t_1, t_2, ∞)-code is a (t_1, t_2)-code.

This paper is organized as follows: In next section, we describe the proposed code construction. Section 3 describes a new class of (t_1, t_2, t_3)-code with tampering detecting capability. The tables and comparisons are shown in Section 4.

2 CODE CONSTRUCTION

The coding approach in this paper needs constant weight codes in [6], [12], [13], [14], [15]. A constant weight codes with minimum Hamming distance 4, 6, 8, 10, can be used as 1,2,3,4-EC/AUED codes, respectively. For example, every codeword in the following 14×8-matrix [15], is a 4-out-of-8 constant weight code with minimum Hamming distance 4, and can be used as 1-EC/AUED code. Since the matrix is used for the tail in this paper, so called as tail matrix.

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

Let the length of (d, k)-constrained sector $\underline{s}$ be n bits. Then, we describe construction of (t_1, t_2)-code as the following:

1) Choose the tail matrix r×l-matrix with rows r_i, $0 \leq i \leq r-1$, of minimum Hamming distance $2(t_2+1)$, and r no less than "$n/(k+1)-n/(d+1)+1$", such that l is as small as possible.
2) Choose row $r_{W(s)-n/(k+1)}$ from r×l-matrix.
3) Use systematic (d, k)-constrained code to encode $r_{W(s)-n/(k+1)}$, and then append it to $\underline{s}$.

The difference of our method and [1] is only to use different types of t_2-EC/AUED codes. So, the proof of that our code is a (t_1, t_2)-code will be same as Theorem 3.1 in [1].

Our method uses a nonsystematic t-EC/AUED code as a tail part instead of systematic t-EC/AUED code. In general, the nonsystematic t-EC/AUED code has better code rate than the systematic t-EC/AUED code for fixed input length. Moreover, the number of codewords needed in our method is just w, where w is the weight span of information part $\underline{s}$, but the method in [1] needs to encode $2^b(\geq w)$ codewords. Due to these two reasons, the tail in our method will be reduced.

Next, we explain how to encode and decode the data. The encoding procedure of our proposed code is shown in Figure 2. The encoding/decoding procedure is shown as follows. Let $\underline{m}$ be an original data, $\underline{u}$ the output n-tuple of (d, k)-constrained RLL code, $\underline{v}$ the output l-tuple of constant weight t-EC/AUED code, $\underline{t}$ the output 2×l-tuple of systematic (d, k)-constrained RLL code, and $\underline{c}$ the t-EC/AUED (n+2×l)-tuple codeword.

Encoding procedure :

1) Encode $\underline{m}$ into $\underline{s}$ in (d, k)-constrained RLL code.
2) Choose row $r_{W(s)-n/(k+1)}$ as $\underline{v}$ form the tail matrix with minimum Hamming distance $2(t_2+1)$.
3) Encode $\underline{v}$ into $\underline{t}$ in systematic (d, k)-constrained RLL code.
4) Output codeword $\underline{c}$=($\underline{s}$, $\underline{t}$).

Decoding procedure :

Let $(\hat{s}, \hat{t})$ be a received codeword.

1) Decode v' from $\hat{t}$ in systematic (d, k)-constrained RLL code..
2) If more than t_2 errors in v', declare tampering and stop. Else, decode v' and get the row i where v' located.
   (NOTE : To correct t_2 errors in the tail part v', non-systematic t-EC/AUED codes can use group theoretical methods with these functions $f_1(\cdot)$, $f_2(\cdot)$, $f_3(\cdot)$, $t_1(\cdot)$, and $t_2(\cdot)$ defined in [11].)
3) If $|W(\hat{s})-(i+n/(k+1))| \leq t_1$, then accept the sector. Otherwise, declare tampering.
4) Decode $\underline{m}$ from $\underline{s}$ in (d, k)-constrained RLL code.

Example 2 : Consider Example 1. Since 4-out-of-8 tail matrix has fourteen rows greater than ten different weights, it can be used as a tail matrix for information part $\underline{s}$

of length 24. Since $W(\underline{s})=7$, and 7-3=4, so we choose row r_4=(10100101), and then encode it to get the 16 bits tail $\underline{t}$ =(0100010000010001). Finally, we get the smaller tail, and save 6 bits when compared to Example 1.

Example 3 : Consider the decoding in Example 2. Assume that the received codeword $(\hat{s}, \hat{t})$ =(111000001001010010001000,0101010000010001). First decode $\hat{t}$ =(0101010000010001) into v'=(11100101). Then, locate the error position and correct it in v' by using group theoretical method [6], [11], [17]. The corrector v' is (10100101) and is the row r_4 in tail matrix. Since $|W(\hat{s})-(i+n/(k+1))|=|8-(4+3)|=1\leq t_1$, then accept the sector.

3 A NEW CLASS OF CODE WITH TAMPERING DETECTING CAPABILITY

In this section, we discuss a more general approach for tampering in a noisy environment. First, review the capabilities of a (t_1, t_2)-code as follows:

1) The code can correct at most t_2 errors, and detect any number of tampering errors in the tail part.
2) If the tampering errors in the tail less or equal than t_2, i.e., we can correct the tampering, then we can detect all the tampering errors in the information part.
3) It is interesting that we can divide the tampering errors in the information part into two classes, tolerate errors (caused by natural errors) and real tampering errors. The number of tolerate errors is t_1, and it can be adaptively changed according to our need.

Consider the tail part of a (t_1, t_2)-code in [1]. If the errors including either natural noise or external tampering do not exceed t_2, we can detect all possible tampering errors in the information part except the t_1 tolerate errors. However, for a certain application, if we just need a code to detect at most t_3 tampering errors instead of all tampering errors. Here, we introduce another new parameter t_3, called as tampering detecting capability, and the code is abbreviated as (t_1, t_2, t_3)-code.

So, a (t_1, t_2, t_3)-code is a code that can be tolerate in the information part at most t_1 normal errors and detect at most t_3 tampering errors, and correct t_2 errors in the tail.

As we know, the weight distribution of a (d, k)-constrained code with length n is between $n/(d+1)$ and $n/(k+1)$. For (t_1, t_2)-code, we use different tails for different weights of (d, k)-constrained codewords. Here, we want to suffer t_1 normal noise and t_3 tampering errors, and hence we can use the same tail to append the codewords which Hamming weight is the congruence *modulo* (t_1+t_3+1). Thus, when no more than t_1 normal noise and t_3 tampering errors occurs, we can detect them. Of course, there are undetectable normal noise and tampering errors, when the error pattern exceeds the capability of (t_1, t_2, t_3)-code.

A coding approach for (t_1, t_2, t_3)-code is given in the following, and the notation is used in the previous section. The encoding/decoding steps are same as the proposed construction, except the step 2 in encoding procedure, and the step 3 in the decoding procedure. We described as follows.

Step2 of Encoding procedure :
We have the following two choices, the proposed construction or the construction in [1].

1. Use the proposed construction. Choose row $r_{(W(s)-n/(k+1))\ mod(t_1+t_3+1)}$ as $\underline{v}$ form the tail matrix with minimum Hamming distance $2(t_2+1)$.
2. Use the construction in [1]. Find the binary representation $\underline{u}$ of "$(W(\underline{s})-n/(k+1))$ $mod(t_1+t_3+1)$", and then use t_2-EC/AUED codes to encode.

Step3 of Decoding procedure :

1. Use the proposed construction. If $|(W(\hat{s})-n/(k+1))$ $mod(t_1+t_3+1)-i|\leq t_1$, then accept the sector. Otherwise, declare tampering.
2. Use the construction in [1]. Decode the correct binary representation $\underline{u}$, and find the corresponding value i. If $|(W(\hat{s})-n/(k+1))$ $mod(t_1+t_3+1)-i|\leq t_1$, then accept the sector. Otherwise, declare tampering.

Example 4 : Consider Example 1. If we want to design a (t_1, t_2, t_3)-code with $t_1=2$, $t_2=1$, $t_3=1$. Use the proposed construction. Since $t_1+t_3+1=2+1+1=4$, thus we need a tail matrix with at least four rows. A 3-out-of-6 tail matrix with minimum Hamming distance 4 shown below has four rows. Since $W(\underline{s})=7$, and 7-3=0 *mod* 4, so we choose row r_0=(100101), and then encode it to get the 12 bits tail $\underline{t}$ =(010000010001). Finally, we get the smaller tail, and save 10 and 4 bits when compared to Example 1 and Example 2, respectively.

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

Example 5 : Consider Example 4, but the sector size=1024 bytes ,i.e., 12288= 1024×8×3/2 total bits, when use code rate 2/3 (1, 7)-constrained code. The length of tail needed in the proposed construction and [1] is 38 and 48, respectively. Because the tail length is a function of the number *modulo* (t_1+t_3+1), so the tail matrix in Example 4 can be used for $(t_1=2, t_2=1, t_3=1)$-code with information part $\underline{s}$ of any length. Finally, we get the length of tail is still 12 bits. The length of tail is significantly reduced due to the compromise of tampering detecting capability. In this case, our code now just can detect one tampering.

Example 6 : Consider the decoding in Example 4. Assume that the received codeword $(\hat{s}, \hat{t})$ = (1110000001001010010001000,010100010001). Decode $\hat{t}$ =(010100010001) into v'=(110101). Then, locate the error position and correct it in v'. The row is r_0 in tail matrix. Since $|(W(\hat{s})-n/(k+1))\ mod(t_1+t_3+1)-i| \leq t_1$,=|(8-3) mod4-0| =1$\leq t_1$, then accept the sector.

The probability that a (t_1, t_2, t_3)-code fails to detect the presence of errors is described as follows. Since undetected errors occur when only there are $a\times(t_1+t_3+1)+b$ 0-errors, where a=1, 2, 3, …, and b=0, 1, …, t_1. For example, in Example 6, if there are 4(or 5) 0-errors in $\hat{s}$,i.e., $W(\hat{s})$=11(or 12), and thus |(11(or 12)-3) mod4-0|=0(or 1)$\leq t_1$ it will cause the undetected errors.

4 TABLES AND COMPARISONS

In this section, our improved proposed code is compared to previously proposed code in [1]. In these comparisons, we use the tail matrix with Hamming weight=4, 6, 8, from TABLE I-A~TABLE-C in [15], and the tail matrix with Hamming weight=10 is from the Sloane's lower bounds for constant weight codes in [14].

The comparisons for (t_1, t_2)-codes, $1 \leqq t_2 \leqq 4$, are listed in Table 1~4. Table 5 shows the length of tail for (t_1, t_2, t_3)-codes with sector size=1024 bytes. In these tables, #i, i=1, 2, denotes the construction in [1], and the proposed construction. In Table 1~5, Δ_1 and Δ_2 are the improvements of tail length which is defined as the tail length in [1] minus the tail length of the proposed code for (t_1, t_2)-codes and (t_1, t_2, t_3)-codes, respectively. In Table 5, the value of Δ_3 is the difference of the tail length of (t_1, t_2)-codes (or we say (t_1, t_2, ∞)-codes) and (t_1, t_2, t_3)-codes using the proposed construction.

We can see that our proposed (t_1, t_2)-codes have smaller tails. In fact, our method results in a saving of 24 bits for some cases. Sector bytes generally tend to be either 512 bytes (6144=512×8×3/2 bits) or 1024 bytes (12288=1024×8×3/2 bits) are shown in the last two rows in Table 4. For example, using our (t_1,t_2=4)-code to encode one 600M byte optical disk, it will save 600M/1024×24 =14,400K bytes for sector=1024 bytes and 600M/512×24 =28,800K bytes for sector=512 bytes. It means that we can save the capacity about ten(twenty) $3 1/2$ -inch floppy disks for sector size 1024(512) bytes.

Consider the (2, 1, ∞)-code in Table 5. The minimum weight of the modulated (1, 7) RLL sequence is 12,288/8=1,536, i.e., 12,288-1536=10,752 0's. It means that the (2, 1, ∞)-code will have tampering detecting capability=10,752 better than (2, 1, 3537)-code. However, (2, 1, 3537)-code saves 2 tail bits for each sector. According the need of system, we can choose the appropriate (t_1, t_2, t_3)-codes, but when the requirement of tampering detecting capability more than 3537, we must choose (2, 1, ∞)-code.

5 CONCLUSION

Our proposed method although uses non-systematic constant weight code, it does not need encoding of constant weight code. However, it needs a table. The table can be easily implemented by a reasonable sized ROM. For example, a (t_1, t_2=4)-code with sector size 1024 bytes in Table 4 only needs a 13×36 ROM (since 13=$\lceil \log_2 4609 \rceil$).

6 REFERENCES

[1] M. Blaum and J. Bruck, "A coding approach for detection of tampering in write-once optical disks", *IEEE trans. computers*, vol. C-47, pp.120-125, Jan. l998.

[2] E.L. Lesis, "Data integrity in digital optical disks", *IEEE trans. Computers*, vol. C-33, pp.818-827, Sep. l984.

[3] 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.

[4] P.H. Siegel, "Recording codes for digital magnetic storage," *IEEE trans. Magnetic*, vol. MAG-21, pp.1344-1349, Sep. l985.

[5] K.A.S. Immink, Coding for digital recorders. Englewood Cliffs, N.J., Prentice Hall, 1991.

[6] B. Bose and T.R.N. Rao, "Theory of unidirectional error correcting/detecting codes," *IEEE trans. Computers*, vol. C-3l, pp.52l-530, June l982.

[7] R.J. McEliece and E.R. Rodemich, "The Constantin-Rao construction for binary asymmetric error-correcting codes," *Inform. contr.*, vol. 44, pp.187-196, l980.

[8] M. Blaum and H. Van Tilborg, "On-*t*-error correcting/all unidirectional error detecting codes," *IEEE Trans. Computers,* vol.38, pp. l493-l50l, Nov. l989.

[9] J. Bruck and M. Blaum, "New techniques for constructing EC/AUED codes," *IEEE Trans. Computers*, vol. C-41, pp.1318~1324, Oct. 1992.

[10] R. Katti and M Blaum, An improvement on constructions of t-EC/AUED codes," *IEEE Trans. Computers*, vol. C-45, pp.607-608, May l996.

[11] S. Al-Bassam and B. Bose, "Design of efficient error-correcting balanced codes," *IEEE Trans. Computers,* vol. C-42, pp.1261-1266, Oct. 1993.

[12] C.S. Laih and C.N. Yang, "On the analysis and design of group theoretical *t*-syEC/AUED codes," *IEEE Trans. Computers*, vol. C-45, pp. 103-108, Jan. 1996.

[13] C.N. Yang and C.S. Laih, "Generating functions for asymmetric/unidirectional error correcting and detecting codes," *IEICE Trans. on Funda.*, vol. C-45, pp. 103-108, Jan. 1997.

[14] R.L. Graham and N.J.A. Sloane, “Lower bounds for constant weight codes,” *IEEE Trans. Information Theory*, vol. IT-26, pp.37-42, Jan. 1980.

[15] A.H. Brouwer, J.B. Shearer, N.J.A. Sloane, and W.D. Smith, “A new table of constant weight codes,” *IEEE Trans. Information Theory*, vol. IT-36, pp.1334-1380, Nov. 1990.

[16] K.A.S. Abdel-Ghaffar and H.C. Ferreira, “Systematic encoding of thr Varshamov-Tenengol’ts codes and the Constantin–Rao codes,” *IEEE Trans. Information Theory*, vol. C-44, pp. 103-108, Jan. 1998.

[17] S. Kundu, “On symmetric error correcting and all unidirectional error detecting codes,” *IEEE Trans. computers*, vol. C-39, pp.752-76l, June l990.

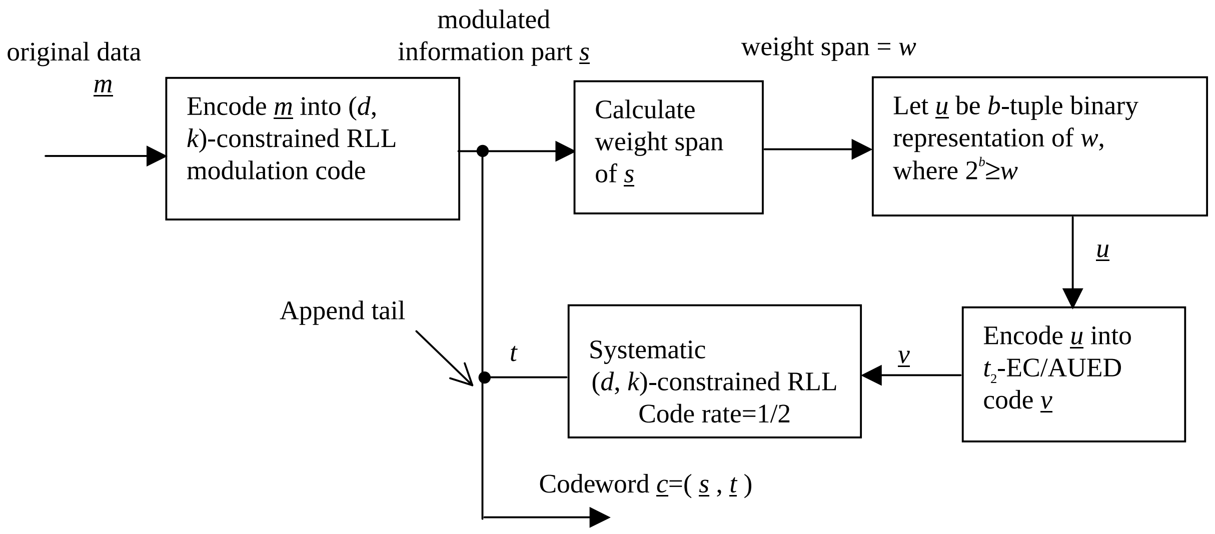
original data $\underline{m}$
modulated information part $\underline{s}$
weight span = w
Encode $\underline{m}$ into (d, k)-constrained RLL modulation code
Calculate weight span of $\underline{s}$
Let $\underline{u}$ be b-tuple binary representation of w, where $2^b \geq w$
$\underline{u}$
Append tail
t
Systematic (d, k)-constrained RLL Code rate=1/2
$\underline{v}$
Encode $\underline{u}$ into t_2-EC/AUED code $\underline{v}$
Codeword $\underline{c}=(\underline{s}, \underline{t})$

Figure 1 : The encoding procedure in [1] for detection of tampering in a noisy environment

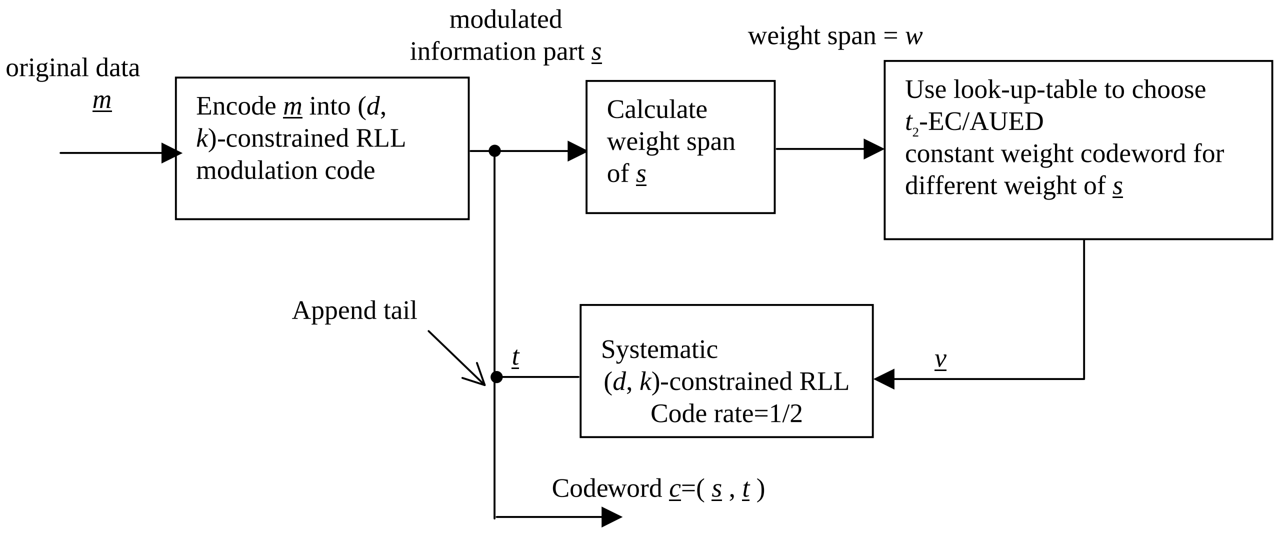
original data $\underline{m}$
modulated information part $\underline{s}$
weight span = w
Encode $\underline{m}$ into (d, k)-constrained RLL modulation code
Calculate weight span of $\underline{s}$
Use look-up-table to choose t_2-EC/AUED constant weight codeword for different weight of $\underline{s}$
Append tail
$\underline{t}$
Systematic (d, k)-constrained RLL Code rate=1/2
$\underline{v}$
Codeword $\underline{c}=(\underline{s}, \underline{t})$

Figure 2 : The proposed encoding procedure for detection of tampering in a noisy environment

Table 1 Comparison of codes with t_2=1

Length of (1,7) code $\underline{s}$	Number of weight span of $\underline{u}$	Bit num. ($b=\lceil \log_2 w \rceil$)	Length of t_2-EC /AUED code $\underline{v}$		Length of tail $\underline{t}$ ($l_i=2\times r_i$)		improvement ($\Delta_1 = l_1 - l_2$)
n	w	b	$v_1^{\#1}$	$v_2^{\#2}$	$l_1^{\#1}$	$l_2^{\#2}$	Δ_1
8	4	2	8	6	16	12	+4
16	7	3	9	7	18	14	+4
24	10	4	12	8	24	16	+8
32	13	4	12	8	24	16	+8
344	130	8	17	12	34	24	+10
12288	4609	13	24	19	48	38	+10

Table 2 Comparison of codes with t_2=2

Length of (1,7) code $\underline{s}$	Number of weight span of $\underline{u}$	Bit num. ($b=\lceil \log_2 w \rceil$)	Length of t_2-EC /AUED code $\underline{v}$		Length of tail $\underline{t}$ ($l_i=2\times v_i$)		improvement ($\Delta_1 = l_1 - l_2$)
n	w	b	$v_1^{\#1}$	$v_2^{\#2}$	$l_1^{\#1}$	$l_2^{\#2}$	Δ_1
24	10	4	17	11	34	22	+12
32	13	4	17	12	34	24	+10
64	25	5	18	13	36	26	+10
104	40	6	19	14	38	28	+10
12288	4609	13	31	24	62	48	+14

Table 3 Comparison of codes with t_2=3

Length of (1,7) code $\underline{s}$	Number of weight span of $\underline{u}$	Bit num. ($b=\lceil \log_2 w \rceil$)	Length of t_2-EC /AUED code $\underline{v}$		Length of tail $\underline{t}$ ($l_i=2\times r_i$)		improvement ($\Delta_1 = l_1 - l_2$)
n	w	b	$v_1^{\#1}$	$v_2^{\#2}$	$l_1^{\#1}$	$l_2^{\#2}$	Δ_1
32	13	4	19	15	38	30	+8
64	25	5	27	16	54	32	+22
120	46	6	28	18	56	36	+20
464	175	8	31	20	62	40	+22
12288	4609	13	37	28	74	56	+18

Table 4 Comparison of codes with t_2=4

Length of (1,7) code $\underline{s}$	Number of weight span of $\underline{u}$	Bit num. ($b=\lceil \log_2 w \rceil$)	Length of t_2-EC /AUED code $\underline{v}$		Length of tail $\underline{t}$ ($l_i=2\times r_i$)		improvement ($\Delta_1 = l_1 - l_2$)
n	w	b	$v_1^{\#1}$	$v_2^{\#2}$	$l_1^{\#1}$	$l_2^{\#2}$	Δ_1
2736	1027	11	46	34	92	68	+24
5464	2050	12	47	35	94	70	+24
6144	2305	12	47	35	94	70	+24
12288	4609	13	48	36	96	72	+24

Table 5 Length of tail for (t_1, t_2, t_3)-codes with sector size=1024 bytes

Parameters of (t_1, t_2, t_3)-code			Length of t_2-EC /AUED code $\underline{v}$		Length of tail $\underline{t}$ (l_i=2×r_i)		improvement (Δ_2= l_1 – l_2)	Δ_3=l_2 of (t_1,t_2,t_3) –l_2 of (t_1,t_2,∞)
t_1	t_2	t_3	$v_1^{\#1}$	$v_2^{\#2}$	$l_1^{\#1}$	$l_2^{\#2}$	Δ_2	Δ_3
2	1	1	8	6	16	12	+4	+26
2	1	129	17	12	34	24	+10	+14
2	1	3537	23	18	46	36	+10	+2
2	1	∞	24	19	48	38	+10	0
3	2	1	16	10	32	20	+12	+28
3	2	116	23	16	46	32	+14	+16
3	2	3581	29	23	58	46	+12	+2
3	2	∞	31	24	62	48	+14	0
4	3	1	18	14	36	28	+8	+28
4	3	171	31	20	62	40	+22	+16
4	3	3918	36	27	72	54	+18	+2
4	3	∞	37	28	74	56	+18	0
5	4	1	31	18	62	36	+26	+36
5	4	57	34	23	68	46	+22	+26
5	4	429	38	28	76	56	+20	+16
5	4	∞	48	36	96	72	+24	0