How to generate cyclically permutable codes from cyclic codes

Kuribayashi, Minoru
Tanaka, Hatsukazu

distributed block lengths, types can naturally be grouped into families [3], where each type in a family occurs with the same probability. For all $i \geq 0, z \geq 1, r \geq i, s \geq i$, $F(i,z,r,s)$ is defined to be the family of types that consist of the following: $2i+1$ blocks, the first of which has length z; the lengths of the i blocks whose bits differ from the first block sum up to s; and the lengths of the remaining i blocks whose bits are the same as the first block sum up to r.

Let K and T be random variables representing the length and type of a block in the received string. Also, let $H(X)$ be the entropy of X measured in bits. Finally, following the notation of [3], note that if we let $\rho_{a,b}$ be the probability that a bits transmitted over a Poisson-repeat channel with parameter λ yield b bits of output, we have

$$\rho_{a,b} = \frac{e^{-\lambda a}(\lambda a)^b}{b!},$$

since the sum of a independent Poisson random variables with mean λ has a Poisson distribution with mean λa. The following lower bound derives immediately from Theorem 4 in [3] (simply by specifying for the Poisson-repeat channel):

Theorem 2: Consider a Poisson-repeat channel with parameter λ and a geometric distribution P with parameter p governing the creation of a random codebook. The capacity of this channel is lower bounded by

$$\sup_{0<p<1} \frac{1}{\frac{1+D}{1-D}\cdot\sum_z zP_z}\left[H(\mathcal{P}) + \sum_k\sum_F\sum_{t\in F}\mathbf{Pr}[T=t,K=k] \cdot \log\left[\frac{d^{r+s+z}\lambda^k}{k!}\cdot\left((r+z)^k - r^k\right)\right]\right] \quad (1)$$

for $d = e^{-\lambda}$, $D = \sum_z P_z d^z$, and F standing for $F(i,z,r,s)$.

Various simplifications can be made to this expression, particularly when choosing codewords governed by block lengths determined by a geometric distribution; the key, however, is that a lower bound for the expression can be calculated numerically. (Truncating the sum appropriately to make it finite will still yield lower bounds, as in [3].)

References

[1] S. Diggavi and M. Grossglauser, "On information transmission over a finite buffer channel," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 1226–1237, Mar. 2006.
[2] E. Drinea and M. Mitzenmacher, "On lower bounds for the capacity of deletion channels," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4648–4657, Oct. 2006.
[3] ——, "Improved lower bounds for i.i.d. deletion and insertion channels," *IEEE Trans. Inf. Theory*, submitted for publication.
[4] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

How to Generate Cyclically Permutable Codes From Cyclic Codes

Minoru Kuribayashi, *Member, IEEE*, and Hatsukazu Tanaka, *Fellow, IEEE*

***Abstract*—On the basis of the characteristics of cyclic codes, the codeword space can be partitioned into small subspaces where cyclically shifted codewords of a particular codeword occupy the same subspace. A cyclically permutable code generates codewords belonging to each subspace. However, no approach for the efficient construction of cyclically permutable code from binary cyclic codes has been proposed thus far. In this study, we propose an approach for the efficient and systematic construction of a cyclically permutable code from a cyclic code by utilizing an algebraic property. The proposed coding method improves the robustness of watermarking, particularly for video frames, against a clipping attack.**

***Index Terms*—Cyclically permutable code, cyclic shift invariant, watermark, clipping attack.**

I. Introduction

A cyclic code [1] is a block code in which a cyclic shift of every codeword yields another codeword belonging to the same code. In this correspondence, a linear cyclic code, which is both linear and closed under cyclic shifting, is employed. Gilbert [2] defined cyclically permutable code (CPC) as a binary block code of block length n such that each codeword has a cyclic order n and the codewords are cyclically distinct (i.e., the same codeword cannot be obtained by cyclically shifting another codeword once more than once). In this scheme, large sets of cyclically permutable codewords are created by interleaving the cyclic shifts of several shorter words and selecting cyclically inequivalent subsets from the resulting set. Maracle and Wolverton [3] proposed an efficient algorithm for generating these cyclically inequivalent subsets. However, in order to use this procedure, sets of cyclically inequivalent placement vectors which indicate the positions of shorter words being interleaved should be selected; consequently, the codeword space is not exploited efficiently. In [4], the correspondence between $m \times n$ arrays and N-tuples, where $N = mn$, is utilized to construct binary constant-weight cyclic codes. This method, when combined with a simple method that selects a large subset of codewords with a full cyclic order, is used to construct a constant-weight CPC. In [5], on the basis of the combinatorial design of a difference family, several constructions for constant-weight CPC are presented. Further, by modifying the recursive constructions for the difference families, other constructions for constant-weight CPC are shown. Although such codes are applied in code-divisible multiple-access (CDMA) communication systems, their procedure for CPC construction is fairly complicated; further, their error correction capability is not discussed. To the best of our knowledge, no approach for efficient and systematic CPC generation directly from binary cyclic codes has been proposed thus far.

Based on the characteristics of cyclic codes, the existence of cyclically equivalent sets can be intuitively determined. Inaba and Nakahara [6] proposed an encoding procedure for obtaining cyclically inequivalent subsets from a cyclic code. However, the scheme only shows an example of the procedure, and the generated code is not a CPC because the cyclic order may be a divisor of the code length.

Manuscript received March 29, 2005; revised June 23, 2006.
M. Kuribayashi is with the Faculty of Engineering, Kobe University, Kobe, 657-8501, Japan (e-mail: kminoru@kobe-u.ac.jp).
H. Tanaka is with the Kobe Institute of Computing, Kobe 650-0001, Japan (e-mail: tanaka@kic.ac.jp).
Communicated by R. J. McEliece, Associate Editor for Coding Theory.
Digital Object Identifier 10.1109/TIT.2006.881834

In this study, we investigate the characteristics of finite fields and cyclic codes and propose an efficient and systematic CPC construction procedure. The proposed approach is based on the partitioning of the codeword space of a cyclic code into cyclically equivalent subsets. In order to simplify the encoding procedure, an information polynomial of an applied cyclic code is developed by exploiting the property of primitive polynomials. By employing error-correcting codes such as the Hamming code, the Bose–Chaudhuri–Hocquenghem (BCH) code [1], etc., the generated CPC inherits the same error-correction capability.

One of the applications of CPC is watermarking [7]. Here, the CPC imparts robustness to a system against clipping attacks. Since the codeword is invariant to cyclic shifts, the encoded watermark information symbols can be extracted from any successive code symbols.

II. Definition of Parameters

Our procedure for CPC construction is based on a binary cyclic code such as the Hamming code or the BCH code. In this section, we review such codes and provide definitions of certain parameters. First, we provide a formal definition of a CPC as follows.

Definition 1: CPC is a set of binary n-tuples such that

$$(c_{i,0}, c_{i,1}, \ldots, c_{i,n-1}) \neq (c_{i,t}, c_{i,t+1}, \ldots, c_{i,n+t-1}), (1 \leq t \leq n-2). \quad (1)$$

For any t

$$(c_{i,t}, c_{i,t+1}, \ldots, c_{i,n+t-1}) \neq (c_{j,0}, c_{j,1}, \ldots, c_{j,n-1}),\ (i \neq j) \quad (2)$$

where the subscripts are modulo-n.

The elements in a finite field GF(2^m) are represented by $\{0, 1, \alpha, \alpha^2, \ldots, \alpha^{n-1}\}$, where $n = 2^m - 1$, and α denotes a root of a primitive polynomial of degree m. Each nonzero element is the root of one of the polynomials that divide $x^n - 1$. Suppose

$$x^n - 1 = (x-1)\prod_{s=1}^{L} g_s(x) \quad (3)$$

where $g_s(x), (1 \leq s \leq L)$, are irreducible polynomials. It is well known that the degree of $g_s(x)$ is a divisor of m. In particular, if n is prime, all the irreducible polynomials are primitive and of degree m. Then, the number of such polynomials is given as follows:

$$L = \frac{n-1}{m}. \quad (4)$$

Let $G(x)$ be a generator polynomial of degree $n-k$ such that $G(x)|x^n - 1$, and let $I(x)$ be an information polynomial of degree k at most. Then, the generated codeword polynomial $C(x)$ is represented as follows:

$$C(x) = I(x)G(x). \quad (5)$$

Since the degree of $C(x)$ is at most n, the codeword is represented by a binary n-tuple.

Since the cyclic order of an n-tuple must be a divisor of n, we can easily obtain a CPC if n is prime. Similarly, the CPC can be defined as a binary block code such that its codewords lie in distinct cyclic equivalence classes and the size or the order of each equivalence class is equal to the block length. In our scheme, such a CPC is constructed from a cyclic code by utilizing the algebraic property. Now, we define the set of elements in each class as an equivalence class.

Definition 2: An equivalence class C is a minimal collection of codeword polynomials such that if $f(x) \in C$ and $g(x) \notin C$

$$\{x^t f(x) | 0 \leq t \leq n-1\} \cap \{x^t g(x)\,|\,0 \leq t \leq n-1\} = \emptyset.$$

Now, we consider the actual CPC construction procedure. The direct brute-force approach is as follows. A codeword of a cyclic code is generated and considered as the first member of the equivalence classes. Then, the second codeword is generated and all its cyclic shifts are compared with the first codeword. If any of the cyclic shifts are identical, the codeword is rejected; otherwise, it is considered as the second member of the equivalence classes. This procedure continues until every codeword has been tested. If the number of equivalence classes is large, such an approach results in an algorithm that is extremely time consuming. Therefore, it is necessary to develop an algorithm for efficient and systematic CPC construction. This correspondence contributes to the development of such an algorithm for binary cyclic codes that utilizes their algebraic properties.

III. Generation of Equivalence Classes

In this section, we illustrate systematic CPC constructions and prove that each codeword occupies a single distinct equivalence class. Here, our aim is to construct an encoder such that its output is a distinct codeword of the CPC for a given input.

Suppose that in GF(2^m), the generator polynomial $G(x)$ is one of the primitive polynomials of degree m, and $g(x)$ is a minimal polynomial with a root α and order equal to the code length $n = 2^m - 1$, where n must be prime in our construction. Let $i(x)$ be a message polynomial. When the following information polynomial is encoded:

$$I(x) = i(x)g(x) + 1 \quad (6)$$

the following codeword is generated:

$$C(x) = \Big(i(x)g(x) + 1\Big)G(x). \quad (7)$$

For a different codeword $C'(x)$ generated from $i'(x)$, each codeword belongs to a distinct equivalence class when the following relation is satisfied:

$$C(x) \neq x^t C'(x),\ 1 \leq t \leq n-2. \quad (8)$$

In (8), when the right-hand side is subtracted from the left-hand side, the following result is obtained:

$$x^t C'(x) - C(x) = \Big(\Big(x^t i'(x) - i(x)\Big)g(x) + (x^t - 1)\Big)G(x). \quad (9)$$

Here, if n is prime, $x^t - 1$ is not a divisor of $x^n - 1$ for any $t < n$; hence, $g(x) \nmid x^t - 1$. Therefore, we get the following relation:

$$x^t C'(x) - C(x) \neq 0 \pmod{x^n - 1}. \quad (10)$$

Based on the above discussion, the following theorem can be proved.

Theorem 1: Suppose $i_j(x)$, where $(1 \leq j \leq L-1)$, is a certain polynomial of a degree less than $k - mj$ and $r(x)$ is one of the polynomials generated as follows:

$$r(x) = x^t,\ (0 \leq t \leq n-2). \quad (11)$$

If the generator polynomial is $G(x) = g_L(x)$ and an information polynomial $I(x)$ satisfies the following equation:

$$I(x) = \Big(i_j(x)g_j(x) + r(x)\Big)\prod_{s=1}^{j-1} g_s(x) \quad (1 \leq j \leq L-1) \quad (12)$$

the generated codeword $C(x)$ belongs to the equivalence class corresponding to $i_j(x)$, where n is prime.

In order to prove Theorem 1, we show that there is no duplication among the generated equivalence classes and they cover all the codewords of the (n, k) cyclic code. It should be noted that the degree of $I(x)$ is less than $k = n - m$ because one primitive polynomial functions as the generator polynomial in this theorem.

Lemma 1: If n is prime and $i_j(x) \neq i'_{j'}(x)$, the generated codewords belong to different equivalence classes.

Proof: The codewords generated from $i_j(x)$ and $i'_{j'}(x)$ are given as follows:

$$C_{i_j}(x) = \Big(i_j(x)g_j(x) + r(x)\Big)\prod_{s=1}^{j-1} g_s(x)g_L(x) \tag{13}$$

$$C_{i'_{j'}}(x) = \Big(i'_{j'}(x)g_{j'}(x) + r(x)\Big)\prod_{s'=1}^{j'-1} g_{s'}(x)g_L(x). \tag{14}$$

We now prove by contradiction that $C_{i_j}(x)$ and $C_{i'_{j'}}(x)$ are in the same equivalence class. Consider that for a nonnegative integer t, a codeword and cyclically t-shifted codeword are identical under modulo $x^n - 1$, i.e.,

$$x^t C_{i_j}(x) = C_{i'_{j'}}(x) \pmod{x^n - 1}. \tag{15}$$

From (13), (14), and (15), if $j \geq j'$, we get

$$\begin{aligned} x^t C_{i_j}(x) - C_{i'_{j'}}(x) &= x^t\Big(i_j(x)g_j(x) + r(x)\Big)\prod_{s=1}^{j-1} g_s(x)g_L(x) \\ &\quad - \Big(i'_{j'}(x)g_{j'}(x) + r(x)\Big)\prod_{s'=1}^{j'-1} g_{s'}(x)g_L(x) \\ &= 0 \pmod{x^n - 1}. \end{aligned} \tag{16}$$

The equation can be represented as follows:

$$\left(\left(x^t i_j(x) - \Big(i'_{j'}(x)g_{j'}(x) + r(x)\Big)\prod_{s'=j+1}^{j'-1} g_{s'}(x)\right)g_j(x) + x^t r(x)\right) \times \prod_{s=1}^{j-1} g_s(x)g_L(x) = 0 \pmod{x^n - 1} \tag{17}$$

If $j = j'$, (16) can be represented as follows:

$$\left(\Big(x^t i_j(x) - i'_{j'}(x)\Big)g_j(x) + (x' - 1)r(x)\right) \times \prod_{s=1}^{j-1} g_s(x)g_L(x) = 0 \pmod{x^n - 1}. \tag{18}$$

The polynomial enclosed within the large parentheses in (17) and (18) must be a multiple of the factor polynomial $(x^n - 1)/\prod_{s=1}^{j-1} g_s(x)g_L(x)$. Here, from (11), it can be observed that the primitive polynomial $g_j(x)$ can divide neither $x^t r(x)$ nor $(x^n - 1)r(x)$. Therefore,

$$\left(x^t i_j(x) - \Big(i'_{j'}(x)g_{i'}(x) + r(x)\Big)\prod_{s'=j+1}^{j'-1} g_{s'}(x)\right)g_j(x) + x^t r(x) \neq B(x)g_j(x) \tag{19}$$

$$\Big(x^t i_j(x) - i'_{j'}(x)\Big)g_j(x) + (x^t - 1)r(x) \neq B(x)g_j(x) \tag{20}$$

where $B(x)$ is a certain polynomial. Based on the above discussion, we conclude that (16) can never be satisfied; hence, the assumption is contradicted. □

In a binary code, the codeword polynomial $C(x)$ is represented as an n-tuple. For simplicity, we denote the all-zero vector corresponding to $C(x) = 0$ by $\mathbf{0}$ and the all one vector corresponding to $C(x) = (x^n - 1)/(x - 1)$ by $\mathbf{1}$.

Lemma 2: If n is prime, the number of elements in an equivalence class is n, except for the two codewords $\mathbf{0}$ and $\mathbf{1}$.

Proof: The codewords $\mathbf{0}$ and $\mathbf{1}$ are cyclically invariant and only one element exists in the equivalence class. If n is prime, $x^n - 1$ is composed of the polynomial $x - 1$ and all the primitive polynomials of degree m. This implies that the generator polynomial $G(x)$ must only comprise such primitive polynomials. Hence, the generated codeword $C(x)$ is distinct from the cyclically shifted codewords $x^t C(x)$, where $(1 \leq t \leq n - 2)$. Since the elements of an equivalence class are expressed as $x^t C(x)$, the lemma is proved. □

For an (n, k) cyclic code, there exist 2^k codewords. Further, based on Lemma 2, $2^k - 2$ codewords exist in their corresponding equivalence classes of cyclic order n. Therefore, the total number S of equivalence classes is calculated as follows:

$$\begin{aligned} S &= \frac{2^k - 2}{n} = \frac{2(2^{k-1} - 1)}{2^m - 1} \\ &= \frac{2(2^m - 1)(2^{k-m-1} + 2^{k-2m-1} + \cdots + 2^m + 1)}{2^m - 1} \\ &= \sum_{j=1}^{(k-1)/m} 2^{m(j-1)+1} = \sum_{j=1}^{L-1} 2^{m(j-1)+1} \end{aligned} \tag{21}$$

From Lemma 1, each codeword $C_j(x)$ generated from each $i_j(x)$ belongs to the corresponding equivalence class. Since the degree of $i_j(x)$ is less than $k - mj$, the number of possible selections of the polynomial $i_j(x)$ is

$$2^{k-m} + 2^{k-2m} + \cdots + 2^{m+1} + 2 = \sum_{j=1}^{L-1} 2^{m(j-1)+1} = S. \tag{22}$$

Hence, all equivalence classes are specified by the codewords generated from $I(x)$ in (12) without duplication. This concludes the proof of Theorem 1. For any generator polynomial of binary cyclic codes, we give the following theorem.

Theorem 2: Let $i_j(x)$, where $(1 \leq j \leq L - \ell)$, be a certain polynomial of a degree less than $k - mj$ and $r(x)$ be one of the polynomials of (11). If a generator polynomial is

$$G(x) = \prod_{j=L-\ell+1}^{L} g_j(x) \tag{23}$$

and the information polynomial $I(x)$ satisfies the following equation:

$$I(x) = \Big(i_j(x)g_j(x) + r(x)\Big)\prod_{s=1}^{j-1} g_s(x) \quad (1 \leq j \leq L - \ell) \tag{24}$$

the generated codeword $C(x)$ belongs to the equivalence class corresponding to $i_j(x)$, where n is prime and $\ell = 1, 2, \ldots, L - 1$.

The proof of the validity of this theorem is entirely analogous to that of Theorem 1; hence, it is omitted.

On the basis of Theorems 1 and 2, we obtain CPC encoder that partitions the codeword space into equivalence classes of the maximum order. Now, we show the actual CPC construction procedure using our method. Let $n = 2^m - 1$ be prime and $k = n - m\ell$ be the degree of the information polynomial. Then, the generator polynomial of the (n, k) cyclic code is represented by (23). By encoding an information polynomial produced in a suitable manner from an information bit string i, a CPC codeword is obtained. Here, S_ℓ which is the number of CPC codewords encoded by using such a procedure is represented by

$$S_\ell = \sum_{j=1}^{L-\ell} 2^{m(j-1)+1}. \tag{25}$$

The detailed procedure for encoding i into the CPC codeword is as follows:

Step 1 An information bit string $i(< S_\ell)$ is represented by the polynomial $i_j(x)$ of degree less than $k - mj$, where

$$j = \begin{cases} 1 & (0 \le i < 2^{k-m}) \\ 2 & (2^{k-m} \le i < 2^{k-m} + 2^{k-2m}) \\ \vdots & \\ L-\ell-1 & (S_\ell - 2^{m+1} \le i < S_\ell - 2) \\ L-\ell & (S_\ell - 2 \le i < S_\ell). \end{cases} \tag{26}$$

Step 2 An information polynomial $I(x)$ is generated from $i_j(x)$ by using (24).

Step 3 $I(x)$ is encoded into the (n, k) code. Then, the result $C(x) = I(x)G(x)$ represents the CPC codeword.

In the brute-force algorithm shown in Section II, it is difficult to choose an arbitrary CPC codeword from an information bit string because the codeword is not assigned. When compared with the brute-force algorithm, our encoding procedure is systematic and convenient because the (n, k) cyclic code encoder can be directly applied. Only the modification of the information polynomial from a given information bit string is required, which is very simple and easy to implement.

IV. Consideration

If $n = 2^m - 1$ is prime, it is called a Mersenne prime; the first 10 Mersenne primes are known to correspond to $m = 2, 3, 5, 7, 13, 17, 19, 31, 61, 89$. In number theory, whether a set of Mersenne primes is finite or infinite is an open question; however, the first 10 primes may be sufficient for CPC construction because of the code length.

An example of our approach for CPC construction is illustrated for $m = 5$. Then, $n = 31$ is prime and

$$x^{31} - 1 = (x-1)g_1(x)g_2(x)g_3(x)g_4(x)g_5(x)g_6(x). \tag{27}$$

If the generator polynomial is selected as

$$G(x) = g_5(x)g_6(x) \tag{28}$$

and the information polynomial as

$$I(x) = \begin{cases} i_1(x)g_1(x) + r(x) \\ \Big(i_2(x)g_2(x) + r(x)\Big)g_1(x) \\ \Big(i_3(x)g_3(x) + r(x)\Big)g_1(x)g_2(x) \\ \Big(i_4(x)g_4(x) + r(x)\Big)g_1(x)g_2(x)g_3(x) \end{cases} \tag{29}$$

the total number of equivalence classes is

$$S_2 = \sum_{j=1}^{4} 2^{5(j-1)+1} = 67650. \tag{30}$$

In the example, a BCH$(31, 21)$ code is applied; therefore, the minimum distance of each codeword is 5. This indicates a double-error correction capability.

One of the important characteristics of a CPC is its cyclically invariant property. If a codeword is iteratively transmitted, the receiver does not need to recover the synchronization. It can decode a message from any successive n symbols. This characteristic may be attractive for several applications that suffer from a synchronization problem.

If the cyclic order is less than n, this characteristic may be inconsequential for some applications that require the invariance property against a cyclic shift. It is known that any cyclic code will be partitioned into small cyclically invariant subspaces. Unfortunately, a general proof for constructing such a subspace from cyclic codes with any code length is still not available; this is a subject for our future work.

V. Implementation for Watermarking System

One of the applications of a CPC is the watermarking system used for movie files. In video watermarking, the watermarking information is iteratively spread over several frames in order to impart robustness against clipping attacks [7]; this is because the recovery from synchronization loss is necessary in watermark extraction. If the watermark information is encoded by using a CPC, it is directly extracted from any of the successive symbols. If the number of symbols is greater than the code length, the embedded watermark information will be decoded with a higher reliability by the error correction assistance of the applied CPC. We assume that the basic cyclic code is a BCH(n, k) code, where $n = 2^m - 1$ is prime and the error correction capability is t. Let $i(x)$ denote an equation that represents the watermark information and $\boldsymbol{i} = (i_0, i_1, i_2, \ldots, i_{k-m-1})$ be its binary representation. It should be noted that the number of elements in $\boldsymbol{i}$ must be less than S_ℓ. First, $\boldsymbol{i}$ is modulated to $\boldsymbol{I} = (I_0, I_1, I_2, \ldots, I_{k-1})$ on the basis of (24). Then, $\boldsymbol{I}$ is encoded as a BCH code and the CPC codeword $\boldsymbol{C} = (c_0, c_1, c_2, \ldots, c_{n-1})$ is obtained. In order to embed it into dozens of frames of a movie file, the codeword $\boldsymbol{C}$ is iteratively arranged.

When the watermark is extracted from any of the successive n frames, each code symbol is extracted from each frame and $\hat{\boldsymbol{C}} = (c_{t_0}, c_{t_1}, c_{t_2}, \ldots, c_{t_{n-1}})$ is obtained. Since the cyclic-shifted codeword still belongs to the same equivalence class, the original codeword is identified. Here, the extracted signal may contain errors due to attacks. If the number of error bits is less than t, the error correction capability, i.e., the exact watermark information can be decoded. In addition, a soft-decision decoding method [8], [9] for error-correction code may be applicable.

VI. Conclusion

We have proposed an approach for the efficient construction of cyclically permutable codes from cyclic codes. Our scheme is applicable to cyclic codes whose code lengths are prime numbers; this scheme can be conveniently implemented because we only add a constraint in the information polynomial of cyclic codes to encode a message. Since the codewords do not change from the original cyclic code, the error correction capability is also inherited. Further, we showed an example of the application of CPC in a watermarking technique. The employment of the code makes the watermark robust against clipping attacks.

Our future work involves the investigation of an efficient decoding algorithm of the CPC from any length of successive symbols, which is greater than the code length.

Acknowledgment

The authors would like to thank the anonymous referees for their helpful comments.

References

[1] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*. Cambridge, MA: MIT Press, 1972.

[2] E. N. Gilbert, "Cyclically permutable error-correcting codes," *IEEE Trans. Inf. Theory*, vol. IT-9, no. 4, pp. 175–182, Jul. 1963.

[3] D. E. Maracle and C. T. Wolverton, "Generating cyclically permutable codes," *IEEE Trans. Inf. Theory*, vol. IT-20, no. 4, pp. 554–555, Jul. 1974.

[4] N. Q. A. L. Gyorfi and J. L. Massey, "Constructions of binary constant-weight cyclic codes and cyclically permutable codes," *IEEE Trans. Inf. Theory*, vol. 38, no. 3, pp. 940–949, May 1992.

[5] S. Bitan and T. Etzion, "Constructions for optimal constant weight cyclically permutable codes and difference families," *IEEE Trans. Inf. Theory*, vol. 41, no. 1, pp. 77–87, Jan. 1995.

[6] H. Inaba and H. Nakahara, "Notes on rotation-resistant digital watermark using radon transform," in *Proc. Int. Symp. Information Theory and Its Applications (ISITA2004)*, Parma, Italy, Oct. 2004, pp. 310–315.

[7] S. Katzenbeisser and F. A. P. Petitcolas, *Information Hiding Techniques for Steganography and Digital Watermarking*. Norwood, MA: Artech House, Jan. 2000.

[8] G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–277, Mar. 1973.

[9] J. K. Wold, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 1, pp. 78–80, Jan. 1978.