X-Code: MDS Array Codes with Optimal Encoding

Lihao Xu and Jehoshua Bruck, *Senior Member, IEEE*

***Abstract*—We present a new class of MDS array codes of size $n \times n$ (n a prime number) called X-code. The X-codes are of *minimum column distance* 3, namely, they can correct either one *column error* or two *column erasures.* The key novelty in X-code is that it has a simple geometrical construction which achieves encoding/update optimal complexity, i.e., a change of any single information bit affects *exactly two* parity bits. The key idea in our constructions is that all parity symbols are placed in *rows* rather than columns.**

***Index Terms*—Array codes, balanced computation, MDS codes, optimal updates, update complexity.**

I. Introduction

Array codes have important applications in communication and storage systems [5], [6], and have been studied extensively [1]–[4], [7]. A common property of these codes is that the encoding and decoding procedures use only simple XOR and cyclic shift operations, thus are more efficient than Reed–Solomon codes in terms of computation complexity [5]. In this correspondence, we present *X-code*, a new class of array codes of size $n \times n$ over any Abelian group $G(q)$ with an addition operation $+$, where q is the size of the group. When $q = 2^m$, the addition operation is just the usual bit-wise XOR operation. Similar to the codes in [1] and [3], the error model of X-code is that errors or erasures are columns of the array, i.e., if one symbol of a column is an error or erasure, then the whole column is considered to be an error or erasure. As usual, the dimension of the code is defined as $k = \log_{q^n} N$, where N is the number of its codewords. Then the code can also be viewed as an (n, k) code over $G(q^n)$. Its distance is also defined over $G(q^n)$, i.e., over the columns of the array. X-code is a maximum distance separable (MDS) code of distance $d = 3$, i.e., $k = n - 2$, which meets the Singleton bound [8]: $d = n - k + 1$.

One important parameter of array codes is the average number of parity bits affected by a change of a single information bit in the codes, called the *update complexity* in this correspondence. This parameter is particularly crucial when the codes are used in storage applications that need frequent updates of information. The codes in [3] use two *dependent* parity columns to make the distance of the codes to be 3. But the dependency between the two parity columns makes update of one information symbol affecting virtually all the parity symbols. So the update complexity of the codes in [3] increases linearly with the number of columns of the array codes, just similar to Reed–Solomon codes. To overcome this drawback, the *EVENODD* codes [1] and their generalizations [2] were designed based on *independent* parity columns resulting in a more efficient information update. The update complexity of *EVENODD* codes approaches 2 as the number of the columns of the codes increases. But it was proven in [2] that for any linear array codes with only parity columns, the update complexity is always *strictly* larger than 2 (the obvious lower bound). Hence, we asked the following question: Is the update complexity of 2 achievable for general array codes? A positive answer to the foregoing question was given a decade ago [9]. The code in [9] was described by its *parity-check matrix* and represented recently in a clearer form, also by a parity-check matrix, in [4]. Here we construct a new family of array codes, called X-codes, which has a simple *geometrical structure* and has an update complexity of *exactly* 2.

Both the X-codes and the codes in [4] and [9] combine information and parity symbols within columns in order to achieve optimal update complexity. The redundancy of X-code is obtained by adding two parity *rows* rather than two parity columns, which results in the nice property that update of one information symbol affects only *two* parity symbols, i.e., the *update complexity* is always 2. In addition, the number of operations for computing parity symbols at every column is the same, namely, the computational load is evenly distributed among all the columns, thus the bottleneck effects of repeated *write* operations are naturally overcome.

The main contribution of this correspondence is constructing X-code, a new class of MDS array codes of distance 3, with the properties of optimal update complexity and balanced computations. The simple geometrical structure of X-code makes its decoding very efficient, for both *two erasures* and *one error*.

This correspondence is organized as follows. In Section II, the encoding scheme of X-code is described, and a proof of its MDS property is presented. In Section III, we provide an efficient decoding algorithm for correcting two erasures, as well as an efficient algorithm for correcting one error. Section IV concludes the correspondence and presents some future research directions.

II. X-Code Description

In X-code, information symbols are placed in an array of size $(n-2) \times n$. Like other array codes [1]–[3], [7], parity symbols are constructed from the information symbols along several *parity-check lines* or *diagonals* of some *slopes* with the addition operation $+$. But instead of being put in separate columns, the parity symbols of the X-code are placed in *two* additional *rows*. So the coded array is of size $n \times n$, with the first $n-2$ rows containing information symbols, and the last two rows containing parity symbols. Notice that each column has information symbols as well as parity symbols, i.e., information symbols and parity symbols, are mixed in each column. Errors or erasures can happen in any column. If an error or an erasure occurs to a symbol in a column, then this column is considered to be an error or erasure column. By the structure of the code, if two columns are erasures, the number of remaining symbols is $n(n-2)$, which is equal to the number of original information symbols, making it possible to recover the two column *erasures*.

A. Encoding Procedure

Let $C_{i,j}$ be the symbol at the ith row and jth column, the parity symbols of X-code are constructed according to the following encoding rules:

$$C_{n-2,i} = \sum_{k=0}^{n-3} C_{k,\langle i+k+2\rangle_n}$$
$$C_{n-1,i} = \sum_{k=0}^{n-3} C_{k,\langle i-k-2\rangle_n} \quad (1)$$

Manuscript received November 12, 1997; revised July 16, 1998. This work was supported in part by the NSF Young Investigator Award CCR-9457811, by the Sloan Research Fellowship, and by DARPA through an agreement with NASA/OSAT.

The authors are with the California Institute of Technology, Pasadena, CA 91125 USA (e-mail: {lihao, bruck}@paradise.caltech.edu).

Communicated by A. Barg, Associate Editor for Coding Theory.
Publisher Item Identifier S 0018-9448(99)00074-7.

where $i = 0, 1, \cdots, n-1$, and $\langle x \rangle_n = x \bmod n$. Geometrically speaking, the two parity rows are just the checksums along diagonals of slopes 1 and -1, respectively.

From the construction of X-code, it is easy to see that the two parity rows are obtained independently, more specifically, each information symbol affects exactly *one* parity symbol in each parity row. All parity symbols only depend on information symbols, but *not* on each other. So updating one information symbol results in updating only *two* parity symbols. Thus X-code has the optimal encoding (or update) property, i.e., it achieves the lower bound 2 of the update complexity for any codes of distance 3.

It is also easy to see that X-code is a cyclic code in terms of columns, i.e., cyclically shifting columns of a codeword of X-code results in another codeword of X-code.

In addition, notice that each column has two parity symbols, each of which is the checksum of $n-2$ information symbols, thus the number of computations (group additions) for parity symbols at each column is $2(n-3)$. This balanced computation property of X-code is very useful in applications that require evenly distributed computations.

B. The MDS Property

In this section, we state and prove the *MDS property* of X-code.

Theorem 1—MDS Property: X-code has column distance of 3, i.e., it is MDS, *if and only if* n is a prime number.

Proof: Let us start with the *sufficient* condition, namely, to prove that for any prime number n, X-code is MDS.

First observe that X-code is a linear code, thus proving that the code has distance of 3 *is equivalent to* proving that the code has *minimum column weight* $w_{\min}$ of 3, i.e., a valid codeword of X-code has at least three nonzero columns. (A column is called a nonzero column if at least one symbol in the column is nonzero.) We will prove it by contradiction.

From the construction of X-code, checksum is obtained along diagonals of slope 1 or slope -1, it is impossible to have only *one* nonzero column, thus $w_{\min} > 1$.

Now suppose $w_{\min} = 2$, then without loss of generality, because of the column cyclic property of X-code, we can assume the nonzero columns are the 0th and kth columns where $1 \le k \le n-1$. Denote the ith symbol of the 0th and kth columns by a_i and b_i, respectively.

Observe that one diagonal of slope 1 or -1 only traverses $n-1$ columns, then among the diagonals of slope 1, the diagonal crossing a_{n-1-k} does *not* cross any symbol of the kth column, and the diagonal crossing b_{k-1} does *not* cross any symbol of the 0th column, so $a_{n-1-k} = 0$ and $b_{k-1} = 0$. Because of the same property of the diagonals of slope -1, we can also get $a_{k-1} = 0$ and $b_{n-1-k} = 0$ (or $b_{n-1} = 0$ if $k = 1$).

Starting from $a_{k-1} = 0$, we get $b_{2k-1} = 0$, since they are in same the diagonal of slope 1; then we get $a_{3k-1} = 0$, since it is on the same diagonal of slope 1 with $b_{2k-1}, \cdots$, and so on, we have

$$a_{k-1} = a_{3k-1} = a_{5k-1} = \cdots = a_{(n-2)k-1} = 0$$

and

$$b_{2k-1} = b_{4k-1} = b_{6k-1} = \cdots = b_{(n-1)k-1} = 0$$

all indices above are $\bmod n$.

Similarly, starting from $a_{n-1-k} = 0$, we have

$$a_{n-1-k} = a_{n-1-3k} = \cdots = a_{n-1-(n-2)k} = 0$$

and

$$b_{n-1-2k} = b_{n-1-4k} = \cdots = b_{n-1-(n-1)k} = 0$$

again, all indices above are $\bmod n$.

We can describe the above four sets of entries in the array as follows. Let

$$A_0 = \{\langle (2m+1)k - 1 \rangle_n : m = 0, 1, \cdots, (n-3)/2\}$$

and

$$A_1 = \{\langle n - (2l+1)k - 1 \rangle_n : l = 0, 1, \cdots, (n-3)/2\}$$

and let

$$B_0 = \{\langle 2mk - 1 \rangle_n : m = 1, 2, \cdots, (n-1)/2\}$$

and

$$B_1 = \{\langle n - 2lk - 1 \rangle_n : l = 1, 2, \cdots, (n-1)/2\}.$$

Notice that all the sets do not include $n-1$, since n is prime. This can also be seen from the construction of X-code, since the $(n-1)$th row is just an imaginary all-0 row and it does not need to be considered. An illustration of the above sets for $n = 5$ and $k = 2$ is as follows:

A_0		B_1		
A_0		B_1		
A_1		B_0		
A_1		B_0		

Since n is prime, for any $1 \le k \le n-1$, $\gcd(n, k) = 1$, $\|A_0\| = \|A_1\| = (n-1)/2$, and if there were such m and l that

$$(2m+1)k - 1 \equiv n - (2l+1)k - 1 \bmod n \tag{2}$$

i.e.,

$$2(m+l+1)k \equiv 0 \bmod n \tag{3}$$

but $1 \le m+l+1 \le n-2$, $\gcd(m+l+1, n) = 1$, $\gcd(2k, n) = 1$, so it is impossible to have such a pair of m and l, i.e., $\|A_0 \cap A_1\| = 0$. Notice that $n-1 \equiv (2[(n-1)/2]+1)k - 1 \bmod n$, we have

$$A_0 \cup A_1 = \{0, 1, \cdots, n-2\}.$$

Similarly,

$$B_0 \cup B_1 = \{0, 1, \cdots, n-2\}.$$

So all the first $n-1$ symbols in the 0th and the kth columns are 0's, obviously the last symbols in the 0th and the kth columns should be also 0's. Thus $w_{\min} \ge 3$, but it is easy to see there is a codeword of column weight 3, so $w_{\min} = 3$. This concludes the proof for the sufficient condition.

On the other hand, from (3), if n were not a prime number, then it could be factored into two factors n_1 and n_2. Thus we got a solution (k, l, m) for (2) or (3), where $k = n_1$ and $m + l + 1 = n_2$, and $2 \le k \le n-1$. This means there is a codeword of weight 2, or the distance of the code is no greater than 2, which contradicts with the fact that the code is of distance 3. So n being a prime number is also a necessary condition to the MDS property of X-code. □

Remarks:

1) For the sufficient condition, we can always find a diagonal of one slope which traverses only one of the two columns. Thus the traversed symbol must be 0. Starting from this 0-symbol, use the diagonal of the other slope crossing this symbol, we can determine that the crossed symbol by the diagonal in the other column must be also 0. So this saw-like recursive procedure

can proceed until it hits a parity symbol at one of the two columns, since a parity symbol can only lie in one diagonal. We call this saw-like recursion a *decoding chain*. Since there are four parity symbols at the two columns, there are at most four decoding chains. (A simple calculation can show that there are two decoding chains when $k = 1$ and four decoding chains otherwise.) The procedure of getting the decoding chains will stop with all the symbols at the two columns as 0's if n is prime. Since this procedure is deterministic once the positions of the two columns are given, it also provides an efficient erasure-decoding algorithm.

2) In the code construction above, we use diagonals of slopes 1 and -1. This choice of slopes is not unique. In fact, codes constructed by the pair of slopes $(s, -s)$, where $s = 1, \cdots, (n-1)/2$, are MDS if and only if n is prime. The proof is similar to the case where the slope pair is $(1, -1)$. It seems that other slope pairs do not provide advantages over $(1, -1)$, so in this correspondence we will focus on X-codes generated by the slope $(1, -1)$.

III. Efficient Decoding Algorithms

In this section, we present decoding algorithms for correcting two erasures or one error of X-code. As the encoding algorithm of the code, decoding algorithms do not require any finite field operations. Instead, the only operations needed are just cyclic shifts and additions, which can be implemented very efficiently with software and/or hardware. It is clear how to correct one erasure, since the erasure can be easily recovered along one of the diagonals. So we will proceed with correcting two erasures.

A. Correcting Two Erasures

First notice that in an array of size $n \times n$, if two columns are erasures, then the basic unknown symbols of the two columns are the information symbols. So the number of unknown symbols is $2(n-2)$. On the other hand, in the remaining array, there are $2(n-2)$ parity symbols which include all the $2(n-2)$ unknown symbols. Hence correcting the two erasures is only a problem of solving $2(n-2)$ unknowns from the $2(n-2)$ linear equations. Since X-code is of distance 3, it can correct two erasures; thus the $2(n-2)$ linear equations must be linearly independent, i.e., the linear equations are solvable. Now notice that a parity symbol can *not* be affected by more than one information symbol in a same column, each equation has at most two unknown symbols, with some having only one unknown symbol. This drastically reduces the complexity of solving the equations.

Suppose the erasure columns are the ith and jth $(0 \le i < j \le n-1)$ columns. Since each diagonal traverses only $n-1$ columns, if a diagonal crosses a column at the last row, no symbols of that column are included in this diagonal. This determines the position of the parity symbol including only one symbol of the two erasure columns, thus this symbol can be immediately recovered from the simple checksum along this diagonal. From this symbol, we can get a decoding chain as discussed in Remark 1 in Section II. Together with the other one (if $j - i = 1$) or three (if $j - i > 1$) decoding chains, all unknown symbols can be recovered.

Now let us calculate the starting parity symbols of the decoding chains. First consider the diagonals of slope 1. Suppose the xth symbol of the ith column is the only unknown symbol in a diagonal, then this diagonal hits the jth column at the $(n-1)$th row, and hits the first parity row at the yth column, i.e., the three points (x, i), $(n-1, j)$, and $(n-2, y)$ are on the same diagonal of slope 1, thus the following equations hold:

$$\begin{cases} (n-1) - x \equiv j - i \bmod n \\ (n-1) - (n-2) \equiv j - y \bmod n. \end{cases}$$

Since $1 \le j - i \le n-1$ and $0 \le j - 1 \le n-2$, the solutions for x and y are

$$\begin{cases} x = \langle (n-1) - (j-i) \rangle_n = (n-1) - (j-i) \\ y = \langle j - 1 \rangle_n = j - 1. \end{cases}$$

So from the parity symbol $C_{n-2,\,j-1}$ we can immediately get the symbol $C_{(n-1)-(j-i),\,i}$ in the ith column. Similarly, the symbol $C_{(j-i)-1,\,j}$ in the jth column can be solved directly from the parity symbol $C_{n-2,\,\langle i-1\rangle_n}$.

Symmetrically with the diagonals of slope -1, the symbol $C_{(j-i)-1,\,i}$ in the ith column can be solved from the parity symbol $C_{n-1,\,\langle j+1\rangle_n}$, and the symbol $C_{(n-1)-(j-i),\,j}$ in the jth column can be solved from the parity symbol $C_{n-1,\,i+1}$.

A formal algorithm for correcting the two erasures ith and jth $(0 \le i < j \le n-1)$ columns of X-code can be described as follows.

Algorithm 1—Correcting Two Erasures: Use each of the four parity symbols

$$C_{n-2,\,j-1},\ C_{n-2,\,\langle i-1\rangle_n},\ C_{n-1,\,\langle j+1\rangle_n},\ \text{and}\ C_{(n-1)-(j-i),\,j}$$

as the starting point of a decoding chain, in each decoding chain use the saw-like recursion to recover unknown symbols until the a parity symbol at one of the two erasure columns is hit, then start a new decoding chain, as discussed in Section II. □

The correctness of the algorithm can be deduced from the proof of Theorem 1 and Remark 1 in Section II. Since solving one unknown symbol needs $(n-3)$ additions, the above algorithm uses $2n(n-3)$ additions to decode two erasure columns, just the same as that of the encoding algorithm.

B. Correcting One Error

To correct one error, the key is to locate the error position. This can be done by computing two *syndrome* vectors from the two parity rows. Since the error is a column error, it is natural to compute the syndromes with respect to *columns* than to *rows* as in the encoding procedure. Once the error location is found, the value of the error can be easily computed along the diagonals of either slope.

Suppose $R = [r_{i,\,j}]_{0 \le i,\, j \le n-1}$ is the error-corrupted array, then construct two arrays

$$U = [u_{i,\,j}]_{0 \le i,\, j \le n-1}$$

and

$$V = [v_{i,\,j}]_{0 \le i,\, j \le n-1}$$

from R, where for $0 \le j \le n-1$

$$u_{i,\,j} = v_{i,\,j} = r_{i,\,j}, \qquad 0 \le i \le n-3 \tag{4}$$

$$u_{n-2,\,j} = r_{n-2,\,j},\ v_{n-2,\,j} = r_{n-1,\,j} \tag{5}$$

$$u_{n-1,\,j} = v_{n-1,\,j} = 0 \tag{6}$$

i.e., U and V are constructed by copying the $n-1$ information rows and parity rows accordingly from R, then adding an imaginary 0-row at the last row. From U and V, compute two *syndrome* vectors S_0 and S_1 as follows:

$$S_0[i] = \sum_{k=0}^{n-1} u_{i+k,\,k} \tag{7}$$

$$S_1[i] = \sum_{k=0}^{n-1} v_{i-k,\,k} \tag{8}$$

all subindices above are $\bmod\ n$.

It is easy to see that the two syndrome vectors are respectively the column checksums along the diagonals of slope 1 and -1, and they should be all-zero vectors if there is no error in the array R. If there is one error in the array R, then the two syndromes are just the cyclic-shifted version of the error vector with respect to the position of the error column, thus its location can be determined simply by cyclic equivalence test which tests if two vectors are equal after cyclic shift of one vector. The following example shows how a single error column is reflected in two syndromes for an X-code of size 5.

Example 1—Syndrome Computation for a 5×5 X-Code: Suppose the third column is an error column, then the two syndrome vectors (S_0 and S_1, respectively) and their corresponding error arrays are as follows:

S_0

0	0	0	e_0	0	e_3
0	0	0	e_1	0	0
0	0	0	e_2	0	e_0
0	0	0	e_3	0	e_1
0	0	0	0	0	e_2

S_1

0	0	0	e_0	0	e_2
0	0	0	e_1	0	e_4
0	0	0	e_2	0	0
0	0	0	e_4	0	e_0
0	0	0	0	0	e_1

So the two syndromes are actually just the original error column vector (cyclic-)shifted in two different directions for the same number of positions. When they are shifted back, then they only differ in at most one position, the number of the positions shifted gives the location of the error column. □

The above example almost gives the decoding algorithm for one error correction. A formal algorithm for correcting one error can be described as follows:

Algorithm II—Correcting One Error: Compute two syndrome vectors S_0 and S_1 from the possibly-error-corrupted array R according to the (4)–(8). If the two syndromes are both all-zero vectors, then there is no error in the array R; otherwise, if there exists such an i that after S_0 cyclically down-shift i positions and S_1 cyclically up-shift i positions their first $n-2$ components are equal and the last components of both are zeros, then the ith column of the array R is an error column. If no such i exists, then there is more than one error column in the array R. □

The correctness proof of the algorithm is as follows:

Proof: To make the proof simpler, some notations are introduced as follows: for a vector V, denote V^T as its transpose; let

$$V = (V[0], V[1], \cdots, V[n-1])^T$$

denote $V^{(1)}$ (or $V^{(-1)}$) as the *down-* (or {*up-*) shifted vector from V, i.e.,

$$V^{(1)} = (V[n-1], V[0], \cdots, V[n-2])^T$$

and

$$V^{(-1)} = (V[1], \cdots, V[n-1], V[0])^T$$

and also

$$V^{(i)} = (V^{(i-1)})^{(1)}$$
$$V^{(-i)} = (V^{-(i-1)})^{(-1)}.$$

If one error occurs at the ith column, and its value is

$$e = (e[0], e[1], \cdots, e[n-2], e[n-1])^T$$

then the two syndromes ((4)–(8)) are

$$S_0 = ((e[0], \cdots, e[n-3], e[n-2], 0)^T)^{(-i)} \quad (9)$$
$$S_1 = ((e[0], \cdots, e[n-3], e[n-1], 0)^T)^{(i)} \quad (10)$$

thus

$$S_0^{(i)} = (e[0], \cdots, e[n-3], e[n-2], 0)^T \quad (11)$$
$$S_1^{(-i)} = (e[0], \cdots, e[n-3], e[n-1], 0)^T. \quad (12)$$

Since X-code can correct one error, which means the location of a single column error can always be found unambiguously, such a unique i can be found that the two shifted syndrome vectors may only differ in the second last component and their last components are both 0's ((11) and (12)). Once the error location i is found, the error value is directly obtained from (11) and (12). □

The above algorithm needs $2n(n-2)$ additions to compute the two syndrome vectors, and on average n cyclic equivalence test operations to get the error location.

IV. Conclusions

We have presented X-code, a new class of $n \times n$ MDS array codes of distance 3. The significant difference of these codes from all other known array codes is that the parity (redundant) symbols are placed in two independent rows rather than columns. Encoding and decoding of the codes may be accomplished using only additions (XOR's). We have proven that n being a prime number is necessary and sufficient for X-code to be MDS. X-code achieves the lower bound of the update complexity for all prime numbers n. It also has balanced computation at each column, which might be very helpful in storage systems and distributed computing systems. Finally, decoding algorithms for correcting erasures and error are given.

One future research problem is to find new MDS codes with optimal update complexity 1) for length of all positive integers rather than only prime numbers, and 2) for distance more than 3. Our preliminary research shows that in general X-code cannot be easily extended to have larger distance by simply using more parity rows and taking more slopes, except for few lengths n. Extended diagonal, i.e., a set of symbols not necessary on a straight line of some slope, may be helpful in extending X-code to have both more general lengths and distances. Further research is still ongoing.

References

[1] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures," *IEEE Trans. Comput.*, vol. 44, pp. 192–202, Feb. 1995.

[2] M. Blaum, J. Bruck, and A. Vardy, "MDS array codes with independent parity Symbols," *IEEE Trans. Inform. Theory*, vol. 42, pp. 529–542, Mar. 1996.

[3] M. Blaum and R. M. Roth, "New array codes for multiple phased burst correction," *IEEE Trans. Inform. Theory*, vol. 39, pp. 66–77, Jan. 1993.

[4] ——, "On lowest density MDS codes," *IEEE Trans. Inform. Theory*, this issue, pp. 000–000..

[5] M. Blaum, P. G. Farrell, and H. C. A. van Tilborg, "Chapter on array codes," in *Handbook of Coding Theory*, V. S. Pless and W. C. Huffman, Eds., to be published.
[6] P. G. Farrell, "A survey of array error control codes," *ETT*, vol. 3, no. 5, pp. 441–454, 1992.
[7] R. M. Goodman, R. J. McEliece, and M. Sayano, "Phased burst error correcting arrays codes," *IEEE Trans. Inform. Theory*, vol. 39, pp. 684–693, 1993.
[8] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.
[9] G. V. Zaitsev, V. A. Zinov'ev, and N. V. Semakov, "Minimum-check-density codes for correctingbytes of errors, erasures, or defects," *Probl. Inform. Transm.*, vol. 19, no. 3, pp. 197–204, 1983.

Weight Hierarchies of Extremal Non-Chain Binary Codes of Dimension 4

Wende Chen and Torleiv Kløve, *Senior Member, IEEE*

***Abstract*—The weight hierarchy of a linear $[n, k; q]$ code C over GF (q) is the sequence $(d_1, d_2, \cdots, d_k)$ where d_r is the smallest support of an r-dimensional subcode of C. An $[n, k; q]$ code is extremal nonchain if, for any r and s, where $1 \leq r < s \leq k$, there are no subspaces D and E such that $D \subset E$, $\dim D = r$, $\dim E = s$, $w_S(D) = d_r$, and $w_S(E) = d_s$. The possible weight hierarchies of such binary codes of dimension 4 are determined.**

***Index Terms*— Binary code, chain condition, difference sequence, support weight, weight hierarchy.**

I. Introduction

The weight hierarchy of linear codes has been studied by a number of researchers. For a code of dimension k, it is a sequence of parameters $(d_1, d_2, \cdots, d_k)$. In particular, d_1 is the minimum distance of the code. The parameters were first introduced in [10]. In [16] it was shown that these parameters are important in the analysis of an application of linear codes to the wiretap channel of type II. Later, the weight hierarchy has been shown to be important in the analysis of the trellis complexity of linear codes, see, e.g., [8], [12], and [15]; and analysis of linear codes for error detection on the local binomial channel, see [14]. The possible weight hierarchies of binary linear codes of dimension up to 4 were determined in [13]. The chain condition was introduced in [17]. Codes satisfying this condition have been studied in, e.g., [1], [8], [9], [12], [15], and [17]. For small lengths and dimensions, the codes with largest values of of the minimum support weights satisfies the chain conditions and this is possibly a general phenomenon. This is the main reason for studying codes satisfying the chain condition. Also, the analysis of the weight hierarchies of product codes is simpler if both codes satisfy the chain condition, see [9] and [17]. The possible weight hierarchies of binary linear codes of dimensions up to 5 satisfying the chain condition were determined in [7]. In [2]–[6] we studied the possible weight hierarchies of linear codes of dimension 4 or less over arbitrary finite fields. The chain condition is a statement that subspaces of smallest support are related in a particular way. To get a better understanding of how weight hierarchies behave in general, it is interesting to study how the subspaces of smallest support are related. One extreme are codes satisfying the chain condition. The other extreme are what we call extremal nonchain codes. In [3] and [4] we determined the possible weight hierarchies of extremal nonchain codes of dimension 3. It turns out the the complexity of doing such a classification increases dramatically with the dimension. In [5] we gave bounds on the weight hierarchies of extremal nonchain codes of dimension 4. In this correspondence we determine exactly the possible weight hierarchies of binary extremal nonchain codes of dimension 4.

Manuscript received October 19, 1997; revised May 26, 1998. This work was supported by The Norwegian Research Council (107542/410) and the National Science Foundation of China (19671087).

W. Chen is with the Laboratory of Systems and Control, Institute of Systems Science, Academia Sinica and Key State Laboratory of Information Security, Graduate School of Academia Sinica, Beijing 100080, China.

T. Kløve is with the Department of Informatics, University of Bergen, N-5020 Bergen, Norway.

Communicated by A. Barg, Associate Editor for Coding Theory.
Publisher Item Identifier S 0018-9448(99)00079-6.

II. Notations and Problem Formulation

Throughout this correspondence, unless otherwise stated, C will be an $[n, 4]$ code, that is, a binary linear code of length n and dimension 4. For convenience we give all definitions below for four-dimensional codes, rather than codes of general dimension, since we concentrate on four-dimensional codes.

For any subcode D of C, we define the *support* of D to be the set of positions where not all the codewords of D are zero, and we denote it by $\chi(D)$. Further, we define the *support weight* of D to be the size of $\chi(D)$, and we denote it by $w_S(D)$.

For $1 \leq r \leq 4$, *the rth minimum support weight* (or generalized Hamming weight) of C is defined by

$$d_r(C) = \min\{w_S(D) | D \text{ is an } [n, r] \text{ subcode of } C\}.$$

The sequence (d_1, d_2, d_3, d_4) is the *weight hierarchy* of C.

We note that if we add a zero-position to an $[n, 4]$ code C we get an $[n+1, 4]$ code

$$C' = l\{(\boldsymbol{c}|0) | \boldsymbol{c} \in \boldsymbol{C}\}.$$

The codes C and C' have the same weight hierarchy. Therefore, without loss of generality, we can restrict ourselves to codes without zero-positions, that is, we will assume that $n = d_4$. Our problems can then be reformulated in terms of projective geometry and we do this next.

The *difference sequence* (DS) (i_0, i_1, i_2, i_3) of a $[d_4, 4]$ code is defined by

$$i_0 = d_4 - d_3, \quad i_1 = d_3 - d_2, \quad i_2 = d_2 - d_1, \quad i_3 = d_1.$$

The difference sequence can easily be computed from the weight hierarchy and *vice versa*.

Let G be a generator matrix for C. For any $\boldsymbol{x} \in \mathrm{GF}(2)^4$, $m(\boldsymbol{x})$, the *value* of $\boldsymbol{x}$ will denote the number of occurrences of $\boldsymbol{x}$ as a column in G. In [11] it was shown that there is a one–one correspondence between the subspaces of C of dimension r and the subspaces of $\mathrm{GF}(2)^4$ of dimension $4 - r$ such that if D corresponds to U, then

$$d_4 - w_S(D) = \sum_{\boldsymbol{x} \in U} m(\boldsymbol{x}).$$

We find it convenient to look at the vectors as projective points.

Let V_3 be the projective space $PG(3, 2)$. A *value assignment* is a function

$$m\colon V_3 \to \mathbb{N} = \{0, 1, 2, \cdots\}.$$