MDS Array Codes with Independent Parity Symbols

Mario Blaum, *Senior Member, IEEE*, Jehoshua Bruck, *Senior Member, IEEE*, and Alexander Vardy, *Senior Member, IEEE*

***Abstract*— A new family of MDS array codes is presented. The code arrays contain p information columns and r independent parity columns, each column consisting of $p-1$ bits, where p is a prime. We extend a previously known construction for the case $r=2$ to three and more parity columns. It is shown that when $r=3$ such extension is possible for any prime p. For larger values of r, we give necessary and sufficient conditions for our codes to be MDS, and then prove that if p belongs to a certain class of primes these conditions are satisfied up to $r \leq 8$. One of the advantages of the new codes is that encoding and decoding may be accomplished using simple cyclic shifts and XOR operations on the columns of the code array. We develop efficient decoding procedures for the case of two- and three-column errors. This again extends the previously known results for the case of a single-column error. Another primary advantage of our codes is related to the problem of efficient information updates. We present upper and lower bounds on the average number of parity bits which have to be updated in an MDS code over GF (2^m), following an update in a single information bit. This average number is of importance in many storage applications which require frequent updates of information. We show that the upper bound obtained from our codes is close to the lower bound and, most importantly, does not depend on the size of the code symbols.**

***Index Terms*— MDS codes, array codes, efficient decoding, efficient information updates.**

I. Introduction

THIS work is concerned with maximum distance separable (MDS) codes, namely, those codes whose minimum Hamming distance attains the Singleton bound for a given length and dimension [10]. The Reed–Solomon (RS) codes are a well-known example of MDS codes. However, with RS codes, a) the encoding and decoding procedures are performed as operations over a finite field, and b) an update in a single information bit requires an update in all the parity symbols and affects a number of bits in each symbol. Thus optimal redundancy is achieved at the expense of additional complexity in the encoding/decoding procedures as well as in the number of parity bits affected by an update in an information bit.

These two properties of RS codes are undesirable for certain channels. First, the fact that encoding/decoding is performed in a finite field makes it unfeasible to use large symbols, since the size of the field grows exponentially with the symbol size. Second, the fact that an update in a single information bit requires to re-compute most of the parity bits is particularly undesirable in storage applications where the stored data has to be frequently updated [12].

In this paper, we present a new family of MDS codes having the following two properties: a) encoding and decoding may be accomplished with simple cyclic shifts and XOR operations on the code symbols, without finite field operations; and b) an update in an information bit affects a minimal number of parity bits. One important application of our codes is in storage systems, such as magnetic tapes and RAID architectures [2]–[6], [12], [14], where the minimal size of the redundant storage, efficient encoding/decoding procedures, as well as the complexity of updating the information are of crucial importance.

The new codes we construct are based on recent work in array codes [1], [2], [4], [7], [8]. We assume that the information is stored in a two-dimensional array of bits. Henceforth we shall identify the symbols of an MDS code with the columns of such an array. Thus the errors that can occur are column errors.

A trivial example of an MDS array code of this type is a simple parity code that can correct a single-column erasure. This code is defined by requiring that the last column in the array is a parity column, given by the exclusive-OR of the other columns. Indeed, this trivial code is MDS, the parity column is computed by simple XOR operations, and an update in a single information bit results in an update in a single parity bit.

The main contribution of this paper is a generalization of this simple code to a family of array codes with the following properties:

P1. The number of parity symbols is one less than the distance—the codes are MDS.

P2. The parity columns may be computed by means of simple XOR operations on the information columns.

P3. Update in an information bit affects, on the average, a minimal number of parity bits.

The first nontrivial generalization of the parity code is the EVENODD code introduced in [2]. The EVENODD code has columns of size $p-1$ for a prime p, and requires two parity symbols. It can correct one error or two erasures. In the next

Manuscript received September 20, 1994; revised November 6, 1995. Part of this work was performed while the authors were with IBM Research Division, Almaden Research Center, San Jose, CA. The research of J. Bruck was supported in part by the NSF Young Investigator Award CCR-9457811, by the Sloan Research Fellowship, and by grants from the IBM Almaden Research Center and the AT&T Foundation. The research of A. Vardy was supported in part under Grant N00014-9610129 from the Joint Services Electronics Program. The material in this paper was presented in part at the IEEE International Symposium on Information Theory, Whistler, BC, Canada, September 1995, and at the 900th Meeting of the American Mathematical Society, Chicago, IL, March 1995.

M. Blaum is with IBM Research Division, Almaden Research Center, San Jose, CA 95120 USA.

J. Bruck is with the California Institute of Technology, Mail Code 136-93, Pasadena, CA 91125 USA.

A. Vardy is with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA.

Publisher Item Identifier S 0018-9448(96)01481-2.

section, we extend the construction of the EVENODD code to a family of codes with r parity symbols, where $r > 2$. By construction, each of the r parity symbols may depend on the information symbols, but *not* on the other parity symbols. This is what we mean by "independent parity symbols" in the title. Note that this *systematic* construction preserves the special structure of the new MDS codes, which makes it possible to avoid finite field operations. In contrast, although the codes of [4] can be also made systematic, this would completely destroy their structure.

In Section II, we show that the new codes are always MDS for $r = 3$. However, if $r > 3$ this is no longer true in general. For larger values of r, we give necessary and sufficient conditions for our codes to be MDS, and present a table of the first few values of p which satisfy these conditions. These conditions are particularly easy to evaluate if p is such that 2 is primitive in GF(p). In fact, we prove that in this case our codes are MDS up to $r \le 5$ for all $p \ne 3$, and up to $r \le 8$ for all $p \notin \{3, 5, 11, 13, 19, 29\}$. In Section III we develop decoding algorithms for the new family of codes, for the case of two and three symbol errors ($r = 4$ and $r = 6$, respectively). Notably, these algorithms do not require finite field operations. Instead, they may be easily implemented using only cyclic shifts and XOR operations on the columns of the error-corrupted array. This, in a sense, extends the algorithms of [4] which are applicable only for the case of a single symbol error. Finally, Section IV deals with upper and lower bounds on the average number of parity bits affected by an update in a single information bit. We show that the upper bound derived by using our codes is not far from the trivial lower bound, and, in fact, the trivial lower bound is unattainable. In particular, for our codes the average number of bits in each parity symbol that are affected by an update in an information bit *does not depend* on the size of the symbols. Thus our codes are indeed suitable for use with very large symbols.

II. A New Family of MDS Array Codes

Let $p \ge 3$ be a prime. We shall deal throughout with $(p-1)\times n$ binary arrays $A = [a_{ij}]$, where a_{ij} is the ith bit in the jth column, for $i = 0, 1, \cdots, p-2$ and $j = 0, 1, \cdots, n-1$. Unless otherwise stated, we assume that $n = p + r$, where r is the number of parity symbols (or columns). This yields the largest possible length for our codes. Analogous results for $n < p + r$ may be immediately obtained by shortening. Write

$$A = (\underline{a}_0, \underline{a}_1, \cdots, \underline{a}_{n-1})$$

where $\underline{a}_0, \underline{a}_1, \cdots, \underline{a}_{n-1} \in \text{GF}(2^{p-1})$ are the columns of A. We shall assume that the columns $\underline{a}_0, \underline{a}_1, \cdots, \underline{a}_{p-1}$ are the information symbols, while the remaining columns are the parity, or redundancy, symbols. The problem, then, is how to compute the redundant part from the information part, so that the properties **P1–P3** are satisfied.

A. Definition of the New Codes

The array codes of [4] were shown therein to be equivalent to the RS codes of length p, with operations taken modulo the polynomial

$$M_p(x) = \frac{x^p - 1}{x - 1} = x^{p-1} + x^{p-2} + \cdots + x + 1.$$

The polynomial $M_p(x)$ is not necessarily irreducible and, hence, these codes are not defined over a field, but rather over the ring of polynomials of degree $\le p-2$ modulo $M_p(x)$.

In terms of the $(p-1)\times n$ array $A = (\underline{a}_0, \underline{a}_1, \cdots, \underline{a}_{n-1})$, we shall view each column $\underline{a}_i$ in the array as a polynomial modulo $M_p(x)$. It is also convenient to assume that the array has an imaginary row of zeros, which makes it a $p\times n$ array. A cyclic shift of a column in such array, that is, a multiplication by x modulo $x^p - 1$, can cause the bit corresponding to the last row to be nonzero. However, in this case, the arithmetic modulo $M_p(x)$ forces to take the complement of the shifted column, restoring the zero in the last position. As in [4], we shall use the notation

$$a(\alpha) = a_{p-2}\alpha^{p-2} + \cdots + a_1\alpha + a_0$$

to denote a polynomial modulo $M_p(x)$. Thus $a(\alpha)b(\alpha)$ denotes polynomial multiplication modulo $M_p(x)$. The usual multiplication of polynomials is written as $a(x)b(x)$. With this notation, we have the following definition.

Definition: A linear code $\mathcal{A}(p, r)$ of length $n = p + r$ and dimension $p = n - r$ over the ring of binary polynomials of degree $\le p - 2$ modulo $M_p(x)$ is defined by:

$$\mathcal{A}(p,r) \overset{\text{def}}{=} \Big\{ A = \Big(a_0(\alpha), a_1(\alpha), \cdots, a_{p-1}(\alpha), \\ a_p(\alpha), a_{p+1}(\alpha), \cdots, a_{p+r-1}(\alpha) \Big) \Big\}$$

where

$$a_{p+j}(\alpha) = \bigoplus_{i=0}^{p-1} \alpha^{ji} a_i(\alpha), \quad \text{for } j = 0, 1, \cdots, r-1 \qquad (1)$$

and $\bigoplus$ stands for summation modulo 2.

Here the first p columns of A are arbitrary information symbols, while the last r columns $a_p(\alpha), a_{p+1}(\alpha), \cdots, a_{p+r-1}(\alpha)$ are the parity symbols. Equation (1) specifies how the parity columns of A should be computed from the information columns. Note that the parity symbols $a_p(\alpha), a_{p+1}(\alpha), \cdots, a_{p+r-1}(\alpha)$ depend on the information symbols, but not on each other. Thus we refer to these parity symbols as *independent*.

The fact that the parity symbols in (1) are independent is the major difference between our codes and those of [4]. This difference is crucial with respect to efficient information updates, or equivalently, with respect to property **P3** specified in the introduction. Referring to (1), it is easy to see that updating a single bit in the information part would in most cases require updating a single bit in each of the parity symbols. In contrast, for Reed–Solomon codes as well as for the array codes introduced in [4], updating a single bit in the information part would usually require updating most of the bits in the redundancy part. This could be quite undesirable for many applications, and in particular for reliable storage of data which requires frequent updates [12].

B. The MDS Property

This subsection is organized as follows. First we prove in Proposition 2.2 that the codes $\mathcal{A}(p,r)$ defined by (1) are indeed MDS for $r \le 3$. The first case where $\mathcal{A}(p,r)$ is not necessarily MDS is for $r = 4$. In Proposition 2.3 we prove a simple necessary and sufficient condition for $\mathcal{A}(p,4)$ to be MDS, and then provide an example where this condition is not satisfied. Next, we generalize the approach of Proposition 2.3 to derive similar necessary and sufficient conditions for increasing values of r. Using these conditions, we compile a table of the first few values of p for which our codes are MDS, up to $r \le 8$. Furthermore, we show in Proposition 2.8 that if $r \le 8$ and $p > 29$ is such that 2 is primitive in GF(p), then the codes $\mathcal{A}(p,r)$ are MDS. The latter property allows us to construct MDS codes over very large alphabets, indeed.

Lemma 2.1 (cf. [4]): Elements of the form α^i and $\alpha^i + \alpha^j$ are invertible modulo $M_p(x)$. In other words, if $i \not\equiv j \pmod p$ then

$$\gcd(x^i, M_p(x)) = \gcd(x^i + x^j, M_p(x)) = 1.$$

Observe that in the ring of polynomials modulo $M_p(x)$, multiplying an arbitrary polynomial $a(\alpha)$ by a power of α corresponds to a cyclic shift, possibly followed by complementation of all the coefficients of $a(\alpha)$. Hence, a systematic parity check matrix for the code $\mathcal{A}(p,r)$, defined in the previous subsection, is given by:

$$H(p,r) \overset{\text{def}}{=} \begin{pmatrix} 1 & 1 & \cdots & 1 & 1 & 0 & \cdots & 0 \\ 1 & \alpha & \cdots & \alpha^{p-1} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{r-1} & \cdots & \alpha^{(r-1)(p-1)} & 0 & 0 & \cdots & 1 \end{pmatrix} \tag{2}$$

Equation (2) along with Lemma 2.1 is all we need to show that our codes are MDS for $r \le 3$.

Proposition 2.2: The minimum distance of $\mathcal{A}(p,r)$ is equal to $r+1$ for $r \le 3$.

Proof: The claim of the proposition is trivially true for $r = 1$. To see that it is true for $r = 2$ note that if exactly one of the information symbols in the array A is nonzero then both parity symbols are nonzero by (1). If there are exactly two nonzero information symbols $a_i(\alpha)$ and $a_j(\alpha)$, then at least one of the parity symbols is nonzero, since $a_p(\alpha) = a_i(\alpha) + a_j(\alpha) = 0$ implies $a_j(\alpha) = a_i(\alpha)$ and hence

$$a_{p+1}(\alpha) = \alpha^i a_i(\alpha) + \alpha^j a_j(\alpha) = (\alpha^i + \alpha^j) a_i(\alpha) \ne 0$$

in view of Lemma 2.1.

It remains to consider the case $r = 3$. We have to show that a square matrix M consisting of any three columns of the parity-check matrix $H(p,3)$ of $\mathcal{A}(p,3)$, given by (2), is nonsingular. Note that if M does not contain any of the last three columns of $H(p,3)$, then it is a Vandermonde matrix whose determinant is given by

$$\det M = \prod_{0 \le j < k \le 2} (\alpha^{i_j} + \alpha^{i_k})$$

where i_0, i_1, i_2 are distinct nonnegative integers $\le p-1$. In view of Lemma 2.1, $\det M \ne 0$ and hence M is nonsingular. If M contains one or more of the last three columns of $H(p,3)$, then expanding the determinant about these columns we again obtain a Vandermonde matrix. In particular this is true for the column $[010]^t$ since for a prime $p \ge 3$ and $i = 0, 1, \cdots, p-1$ the third row entries α^{2i} are all distinct. □

Note that the proof of Proposition 2.2 is quite similar to the proof of the well-known fact that triply extended Reed–Solomon codes are MDS [10, p. 326]. It is also well known that it is not possible to further extend the Reed–Solomon codes in this way. However, the sequence of elements $1, \alpha, \alpha^2, \cdots, \alpha^{p-1}$ in the ring of binary polynomials modulo $M_p(x)$ is quite different from the sequence of elements $1, \beta, \beta^2, \cdots, \beta^{q-1}$ in the field GF(q) = GF(2^{p-1}), where β is primitive in GF(q). Indeed, $1, \alpha, \alpha^2, \cdots, \alpha^{p-1}$ contains only a small fraction of all the elements of the ring. Thus we have the following proposition.

Proposition 2.3: The code $\mathcal{A}(p,4)$ is MDS if and only if the polynomial $1 + x^{k_1} + x^{k_2}$ has no common factors with $M_p(x)$, that is

$$\gcd(M_p(x), 1 + x^{k_1} + x^{k_2}) = 1$$

for all $1 \le k_1 < k_2 \le p-1$.

Proof: According to (2), the parity check matrix for $\mathcal{A}(p,4)$ is

$$H(p,4) = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & 0 & 0 & 0 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{p-1} & 0 & 1 & 0 & 0 \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(p-1)} & 0 & 0 & 1 & 0 \\ 1 & \alpha^3 & \alpha^6 & \cdots & \alpha^{3(p-1)} & 0 & 0 & 0 & 1 \end{bmatrix}.$$

We need to consider a square matrix consisting of any four columns of $H(p,4)$. It may be readily seen, using arguments similar to those of Proposition 2.2, that such a matrix cannot be singular unless it contains one of the columns $[0100]^t$ or $[0010]^t$ but not both. This leads to the following two cases:

$$M_1 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ \alpha^{i_0} & \alpha^{i_1} & \alpha^{i_2} & 1 \\ \alpha^{2i_0} & \alpha^{2i_1} & \alpha^{2i_2} & 0 \\ \alpha^{3i_0} & \alpha^{3i_1} & \alpha^{3i_2} & 0 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ \alpha^{i_0} & \alpha^{i_1} & \alpha^{i_2} & 0 \\ \alpha^{2i_0} & \alpha^{2i_1} & \alpha^{2i_2} & 1 \\ \alpha^{3i_0} & \alpha^{3i_1} & \alpha^{3i_2} & 0 \end{bmatrix}.$$

We have

$$\det M_1 = (\alpha^{i_0} + \alpha^{i_1})(\alpha^{i_0} + \alpha^{i_2})(\alpha^{i_1} + \alpha^{i_2}) \cdot (\alpha^{i_0+i_1} + \alpha^{i_0+i_2} + \alpha^{i_1+i_2}) \tag{3}$$

$$\det M_2 = (\alpha^{i_0} + \alpha^{i_1})(\alpha^{i_0} + \alpha^{i_2})(\alpha^{i_1} + \alpha^{i_2}) \cdot (\alpha^{i_0} + \alpha^{i_1} + \alpha^{i_2}). \tag{4}$$

By Lemma 2.1, the first three factors in both (3) and (4) are invertible in the ring of polynomials modulo $M_p(x)$. Thus the matrices M_1 and M_2 are nonsingular if and only if the polynomials $f_1(x) = x^{i_0+i_1} + x^{i_0+i_2} + x^{i_1+i_2}$ and $f_2(x) = x^{i_0} + x^{i_1} + x^{i_2}$ have no common factors with $M_p(x)$. Without loss of generality, assume that $i_0 < i_1 < i_2$. Further, set $k_1 = i_1 - i_0$ and $k_2 = i_2 - i_0$, in which case $1 \leq k_1 < k_2 \leq p-1$. Then $f_1(x) = x^{i_0+i_1}(1 + x^{k_2-k_1} + x^{k_2})$ and $f_2(x) = x^{i_0}(1+x^{k_1}+x^{k_2})$. Yet when k_1 and k_2 vary through all the possible values both $1+x^{k_1}+x^{k_2}$ and $1+x^{k_2-k_1}+x^{k_2}$ range through the same polynomials. □

The condition of Proposition 2.3 is particularly easy to evaluate when p is such that 2 is primitive in GF(p), as exhibited by the following corollary.

Corollary 2.4: If 2 is a primitive element in GF(p) and $p \neq 3$, then $\mathcal{A}(p,4)$ is MDS.

Proof: In this case $M_p(x)$ is irreducible over GF(2) (cf. [10, p. 197]). Thus, $\gcd(M_p(x), 1 + x^{k_1} + x^{k_2}) \neq 1$ if and only if $1 + x^{k_1} + x^{k_2}$ is divisible by $M_p(x)$. Since $1 \leq k_1 < k_2 \leq p-1$, this only happens for $p = 3$. □

What happens for those primes for which 2 is not primitive in GF(p)? For some of them, $\mathcal{A}(p,4)$ still has minimum distance 5. However, for certain values of p, there exist integers $1 \leq i_1 < i_2 \leq p-1$ such that

$$\gcd(1 + x^{i_1} + x^{i_2}, M_p(x)) \neq 1.$$

In these cases, according to the proof of Proposition 2.3, $\mathcal{A}(p,4)$ contains a codeword of weight 4 whose nonzero entries are in locations $0, i_1, i_2$, and $p+2$. For example, for $p = 7$, the following is a codeword of weight 4 in $\mathcal{A}(7,4)$:

0	1	0	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	1	0
1	1	0	0	0	0	0	0	0	0	0

It is easy to see how the argument of Proposition 2.3 may be extended for values of r greater than 4. Obviously $\mathcal{A}(p,r)$ is MDS if and only if every $r \times r$ submatrix of $H(p,r)$ is nonsingular. Any such submatrix M corresponds to a binary vector $\underline{u}$ of length $p+r$ and weight r, where the nonzero entries of $\underline{u}$ indicate the columns of $H(p,r)$ contained in M. Let us write $\underline{u} = (\underline{u}'|\underline{v})$, where $\underline{v}$ is of length r and $(\cdot|\cdot)$ denotes concatenation. Clearly, the indicator vector $\underline{v}$ corresponds to the redundant part of the codeword. Hence if $\underline{v} = \underline{0}$ then M is a Vandermonde matrix and is therefore nonsingular. Thus we need to consider the $2^r - 1$ classes of polynomials corresponding to the nonzero values of $\underline{v}$. For each such value of $\underline{v}$ we may compute the determinant $g(\alpha)$ of the corresponding matrix M and then check whether the polynomials $g(x)$ and $M_p(x)$ have common factors.

In fact, the number of different classes of polynomials $g(x)$ that we actually need to check is much less than $2^r - 1$. As we have seen in Proposition 2.3, for $r = 4$ we only need to consider one polynomial rather than $2^4 - 1 = 15$. In general, we have

Theorem 2.5: The total number $\nu(r)$ of different classes of polynomials that have to be checked, in order to establish whether $\mathcal{A}(p,r)$ is MDS, is upper-bounded by

$$\nu(r) \leq 2^{r-3} + 2^{\lceil (r-2)/2 \rceil - 1} - \tau(r-1) - 1 \qquad (5)$$

where $\tau(n)$ denotes the number of divisors of n different from 1.

The proof of Theorem 2.5 is deferred to the Appendix. Indeed, the upper bound of Theorem 2.5 is still exponential in r. However, since we are primarily interested in small values of r corresponding to the high-rate codes, the difference between (5) and $2^r - 1$ is quite significant. In particular, (5) yields the following upper bounds on the first few values of $\nu(r)$:

r	4	5	6	7	8
$\nu(r)$	1	3	8	16	34

The resulting polynomials $g(x)$ are shown in Table II for $r = 4, 5, 6$. These polynomials were obtained by computing the determinant of the corresponding submatrices of $H(p,r)$, and then factoring out those terms that are invertible modulo $M_p(x)$ by Lemma 2.1.

Once all the polynomials $g(x)$ have been computed, it may be checked by direct computer search whether there exists an assignment of values for $k_1, k_2, \cdots, k_{m-1}$ such that $\gcd(g(x), M_p(x)) \neq 1$ for a given prime p. It follows from the foregoing discussion that the code $\mathcal{A}(p,r)$ is MDS if and only if no such assignment exists for all the $\nu(r)$ polynomials $g(x)$ corresponding to the given value of r. Some of the values of r and p for which the codes $\mathcal{A}(p,r)$ were found to be MDS using this procedure are listed in Table I.

Indeed, as we have seen in Corollary 2.4, the condition $\gcd(g(x), M_p(x)) \neq 1$ is particularly easy to evaluate if $M_p(x)$ is irreducible over GF(2), which happens if and only if 2 is a primitive element in GF(p). Thus we have the following generalization of Corollary 2.4.

Theorem 2.6: If 2 is a primitive element in GF(p), then:

a) The codes $\mathcal{A}(p,4)$ and $\mathcal{A}(p,5)$ are MDS for all $p \neq 3$.
b) The codes $\mathcal{A}(p,6)$ are MDS for all $p \neq 3, 5, 13$.
c) The codes $\mathcal{A}(p,7)$ are MDS for all $p \neq 3, 5, 11, 13$.
d) The codes $\mathcal{A}(p,8)$ are MDS for all $p \neq 3, 5, 11, 13, 19, 29$.

Proof: We illustrate the proof for the case $r = 5$. The three polynomials $g_1(x), g_2(x), g_3(x)$ that we need to consider are given in Table I. Note that each of these polynomials has an even number of terms and, hence, is divisible by $x - 1$. Since $M_p(x)$ is irreducible and $(x-1)M_p(x) = x^p - 1$, it follows that $\gcd(g_i(x), M_p(x)) \neq 1$ if and only if $g_i(x) \equiv 0 \bmod (x^p - 1)$. To evaluate $g_i(x)$ modulo $x^p - 1$ it would suffice to take the exponents of the terms in $g_i(x)$ modulo p. Thus clearly

$$g_2(x) = 1 + x^{k_1} + x^{k_2} + x^{k_3} \not\equiv 0 \bmod (x^p - 1).$$

Further

$$g_1(x) = 1 + x^{k_1} + x^{k_2} + x^{2k_1} + x^{2k_2} + x^{k_1+k_2} \equiv 0 \bmod (x^p - 1)$$

if and only if the set of exponents $\{0, k_1, k_2, 2k_1, 2k_2, k_1 + k_2\}$ may be partitioned into three pairs $\{a, b\}, \{c, d\}$, and $\{e, f\}$ such that

$$a \equiv b \qquad c \equiv d \qquad e \equiv f \qquad (\bmod p).$$

Any such partition leads to a system of equations in k_1, k_2, k_3 modulo p. It is easy to see that none of these systems of

TABLE I
SOME VALUES OF p AND r FOR WHICH $\mathcal{A}(p,r)$ IS MDS
o—The code is MDS ; •—The code is not MDS
Those primes p for which $M_p(x)$ is irreducible are set in boldface.

Symbol size	Redundancy				
p	$r=4$	$r=5$	$r=6$	$r=7$	$r=8$
3	●	●	●	●	●
5	○	○	●	●	●
7	●	●	●	●	●
11	○	○	○	●	●
13	○	○	●	●	●
17	○	●	●	●	●
19	○	○	○	○	●
23	○	○	○	●	●
29	○	○	○	○	●
31	●	●	●	●	●
37	○	○	○	○	○
41	○	○	○	●	●
43	○	●	●	●	●
47	○	○	○	○	○
53	○	○	○	○	○
59	○	○	○	○	○
61	○	○	○	○	○
67	○	○	○	○	○
71	○	○	○	○	●
73	●	●	●	●	●
79	○	○	○	○	
83	○	○	○	○	○
89	○	●	●	●	●
97	○	○	○	○	
101	○	○	○	○	○
103	○	○	○	○	
107	○	○	○	○	○
109	○	○	○	●	●
113	○	○	○	●	●
127	●	●	●	●	●

equations has a feasible solution, unless $p=3$. For instance

$$\begin{cases} k_1+k_2 & \equiv & 0 \\ 2k_2 & \equiv & k_1 \\ 2k_1 & \equiv & k_2 \end{cases}$$

leads to $3k_1 \equiv 0 \pmod{p}$, which is only possible if $p=3$. Similarly, for

$$g_3(x) = x^{k_1} + x^{k_2} + x^{k_3} + x^{k_1+k_2} + x^{k_1+k_3} + x^{k_2+k_3}$$

we obtain

$$\begin{cases} k_2+k_3 & \equiv & k_1 \\ k_1+k_3 & \equiv & k_2 \\ k_1+k_2 & \equiv & k_3 \end{cases}$$

implying $2k_1 \equiv 0 \pmod{p}$, which is a contradiction. This establishes part a). To prove b), c), and d) we enumerate for each $r=6,7,8$ over the $\nu(r)$ test polynomials $g_i(x)$ corresponding to the given value of r. If $g_i(x)$ was obtained by computing a determinant of order m and contains an even number t of terms, then the complexity of checking all the possible partitions of the set of exponents in $g_i(x)$ is $O(t^{m-1})$. Referring to Table III for the values of m and t, we see that this may be done in a reasonable time on a contemporary computer.

It remains to deal with those cases where the number of terms in $g_i(x)$ is odd. These cases may be settled using the following lemma.

Lemma 2.7: If $g(x)$ has an odd number t of terms and $p>t$ then $\gcd(g(x), M_p(x)) = 1$, provided $M_p(x)$ is irreducible.

Proof: Evidently

$$x^n \bmod M_p(x) \;=\; x^n \bmod (x^p-1)$$

unless n is of the form $qp-1$, in which case

$$x^n \equiv x^{p-2} + x^{p-3} + \cdots + x + 1 \mod M_p(x).$$

Now let $g'(x)$ be the polynomial consisting of all those terms in $g(x)$ that have the form x^{qp-1} for some q, and let $g''(x) = g(x) - g'(x)$. Note that, by the foregoing argument, evaluating $g''(x)$ modulo $M_p(x)$ is the same as evaluating $g''(x)$ modulo (x^p-1). Furthermore, $g(x) \equiv 0 \bmod M_p(x)$ if and only if

$$g''(x) \equiv g'(x) \bmod M_p(x). \tag{6}$$

If the number of terms in $g'(x)$ is even, then $g'(x) \equiv 0 \bmod M_p(x)$ and (6) becomes

$$g''(x) \equiv 0 \bmod (x^p-1). \tag{7}$$

However, (7) implies that $(x-1)$ divides $g''(x)$ which is impossible since the number of terms in $g''(x)$ is odd in this case. Hence the number of terms in $g'(x)$ must be odd and (6) becomes

$$g''(x) \equiv x^{p-2} + x^{p-3} + \cdots + x + 1 \bmod (x^p-1). \tag{8}$$

Clearly, (8) can be satisfied only if $g''(x)$ contains at least $p-1$ terms. But then the total number of terms in $g(x)$ must be at least $(p-1)+1=p$. □

It follows from Lemma 2.7 in conjunction with Table III that we only need to check the test polynomials with an odd number of terms up to $p \le 121$. Referring to Table I completes the proof of the theorem. □

We note that it would be hardly possible to extend the method of proof of Theorem 2.6 for values of r significantly greater than 8, since the complexity of the proof increases exponentially with r. However, it is the low-redundancy MDS codes corresponding to the first few values of r that are of importance for most applications [3], [6], [12], [14]. The result of Theorem 2.6 enables us to easily construct low-redundancy MDS codes $\mathcal{A}(p,r)$ for very large values of the symbol-alphabet size p.

III. DECODING ALGORITHMS

In this section we present decoding methods for correcting two- and three-symbol errors using the MDS array codes of Section II. Notably, the proposed decoders do not require finite field operations. Instead, the only operations performed

TABLE II
TEST POLYNOMIALS FOR $r = 4, 5, 6$.

Four parity columns ($r = 4$)	
$g_1(x)$	$= 1 + x^{k_1} + x^{k_2}$
Five parity columns ($r = 5$)	
$g_1(x)$	$= 1 + x^{k_1} + x^{k_2} + x^{2k_1} + x^{2k_2} + x^{k_1+k_2}$
$g_2(x)$	$= 1 + x^{k_1} + x^{k_2} + x^{k_3}$
$g_3(x)$	$= x^{k_1} + x^{k_2} + x^{k_3} + x^{k_1+k_2} + x^{k_1+k_3} + x^{k_2+k_3}$
Six parity columns ($r = 6$)	
$g_1(x)$	$= 1 + x^{k_1} + x^{2k_1} + x^{3k_1} + x^{k_2} + x^{2k_2} + x^{3k_2} + x^{k_1+k_2} + x^{k_1+2k_2} + x^{2k_1+k_2}$
$g_2(x)$	$= x^{k_1} + x^{2k_1} + x^{3k_1} + x^{k_2} + x^{2k_2} + x^{3k_2} + x^{k_1+3k_2} + x^{2k_1+2k_2} + x^{3k_1+k_2}$
$g_3(x)$	$= 1 + x^{k_1} + x^{2k_1} + x^{k_2} + x^{2k_2} + x^{k_3} + x^{2k_3} + x^{k_1+k_2} + x^{k_1+k_3} + x^{k_2+k_3}$
$g_4(x)$	$= x^{k_1} + x^{2k_1} + x^{k_2} + x^{2k_2} + x^{k_3} + x^{2k_3} + x^{k_1+2k_2}\ x^{k_1+2k_3} + x^{2k_1+k_2} + x^{2k_1+k_3} + x^{k_2+2k_3} + x^{2k_2+k_3}$
$g_5(x)$	$= x^{2k_1} + x^{2k_2} + x^{2k_3} + x^{k_1+k_2} + x^{k_1+2k_2} + x^{k_1+k_3} + x^{k_1+2k_3} + x^{2k_1+k_2} + x^{2k_1+2k_2} + x^{2k_1+k_3} + x^{2k_1+2k_3} + x^{k_2+k_3} + x^{k_2+2k_3} + x^{2k_2+k_3} + x^{2k_2+2k_3} + x^{k_1+k_2+2k_3} + x^{k_1+2k_2+k_3} + x^{2k_1+k_2+k_3}$
$g_6(x)$	$= x^{k_1+k_2} + x^{k_1+2k_2} + x^{k_1+k_3} + x^{k_1+2k_3} + x^{2k_1+k_2} + x^{2k_1+k_3} + x^{k_2+k_3} + x^{k_2+2k_3} + x^{2k_2+k_3} + x^{k_1+k_2+k_3} + x^{k_1+k_2+2k_3} + x^{k_1+2k_2+k_3} + x^{2k_1+k_2+k_3}$
$g_7(x)$	$= 1 + x^{k_1} + x^{k_2} + x^{k_3} + x^{k_4}$
$g_8(x)$	$= x^{k_1} + x^{k_2} + x^{k_3} + x^{k_4} + x^{k_1+k_2} + x^{k_1+k_3} + x^{k_1+k_4} + x^{k_2+k_3} + x^{k_2+k_4} + x^{k_3+k_4}$

in these decoders are cyclic shifts, bitwise exclusive-OR, and the operation of testing for cyclic equivalence between two binary vectors. In terms of the number of such operations, the decoding complexity is $O(p)$ for two errors and $O(p^2)$ for three errors.

Before we proceed with decoding two and three errors with $\mathcal{A}(p,4)$ and $\mathcal{A}(p,6)$, respectively, let us briefly consider correcting a single error with $\mathcal{A}(p,2)$. For this case, the parity-check matrix, as defined by (2), is

$$H(p,2) = \begin{pmatrix} 1 & 1 & \cdots & 1 & 1 & 0 \\ 1 & \alpha & \cdots & \alpha^{p-1} & 0 & 1 \end{pmatrix}. \tag{9}$$

Let $B = [b_{ij}]$ be the error-corrupted array. Write

$$B = (b_0(\alpha), b_1(\alpha), \cdots, b_{p+1}(\alpha))$$

and assume that at most one column is in error. Now, define the horizontal and diagonal syndromes by

$$s_0(\alpha) = \bigoplus_{l=0}^{p} b_l(\alpha) \tag{10}$$

$$s_1(\alpha) = b_{p+1}(\alpha) \oplus \left(\bigoplus_{l=0}^{p-1} \alpha^l b_l(\alpha) \right). \tag{11}$$

Assuming that the jth column is in error, and that the error value is $e(\alpha)$, we obtain

$$s_0(\alpha) = e(\alpha) \tag{12}$$

$$s_1(\alpha) = \alpha^j e(\alpha) \tag{13}$$

provided $0 \le j \le p-1$. Combining (12) and (13), we have

$$\alpha^j s_0(\alpha) = s_1(\alpha). \tag{14}$$

Hence, the location in error is given by the first integer j satisfying (14). If no such j exists and one of $s_0(\alpha)$, $s_1(\alpha)$ is nonzero, then there is an error in the corresponding parity column.

A. Correcting Two Symbols in Error

We start with some notation. As in (10) and (11), define the jth syndrome with respect to the parity-check matrix (2) by

$$s_j(\alpha) \overset{\text{def}}{=} b_{p+j}(\alpha) \oplus \left(\bigoplus_{l=0}^{p-1} \alpha^{jl} b_l(\alpha) \right) \quad \text{for } j = 0, 1, \cdots, r-1 \tag{15}$$

TABLE III
THE ORDER m AND THE NUMBER OF TERMS t IN THE TEST POLYNOMIALS FOR $r = 4,5,6,7,8$
The polynomials are represented by the set of zero positions in $\underline{v}$ ($a_1 = 0, a_2, \cdots, a_{m-1}, a_m = r-1$).

r	polynomial	m	t
4	(0,1,3)	3	3
5	(0,1,4)	3	6
	(0,1,2,4)	4	4
	(0,1,3,4)	4	6
6	(0,1,5)	3	10
	(0,2,5)	3	9
	(0,1,2,5)	4	10
	(0,1,3,5)	4	12
	(0,1,4,5)	4	18
	(0,2,3,5)	4	13
	(0,1,2,3,5)	5	5
	(0,1,2,4,5)	5	10
7	(0,1,6)	3	15
	(0,2,6)	3	3
	(0,1,2,6)	4	20
	(0,1,3,6)	4	19
	(0,1,4,6)	4	28
	(0,1,5,6)	4	38
	(0,2,3,6)	4	28
	(0,1,2,3,6)	5	15
	(0,1,2,4,6)	5	20
	(0,1,2,5,6)	5	40
	(0,1,3,4,6)	5	35
	(0,1,3,5,6)	5	31
	(0,2,3,4,6)	5	20
	(0,1,2,3,4,6)	6	6
	(0,1,2,3,5,6)	6	15
	(0,1,2,4,5,6)	6	20
8	(0,1,7)	3	21
	(0,2,7)	3	15
	(0,3,7)	3	18
	(0,1,2,7)	4	35
	(0,1,3,7)	4	28
	(0,1,4,7)	4	38
	(0,1,5,7)	4	52
	(0,1,6,7)	4	67
	(0,2,3,7)	4	50
	(0,2,4,7)	4	36
	(0,2,5,7)	4	49
	(0,3,4,7)	4	54
	(0,1,2,3,7)	5	35
	(0,1,2,4,7)	5	35
	(0,1,2,5,7)	5	71
	(0,1,2,6,7)	5	105
	(0,1,3,4,7)	5	80
	(0,1,3,5,7)	5	60
	(0,1,3,6,7)	5	85
	(0,2,3,4,7)	5	120
	(0,2,3,5,7)	5	50
	(0,2,3,6,7)	5	65
	(0,1,2,3,4,7)	6	21
	(0,1,2,3,5,7)	6	30
	(0,1,2,3,6,7)	6	75
	(0,1,2,4,5,7)	6	75
	(0,1,2,4,6,7)	6	66
	(0,1,2,5,6,7)	6	111
	(0,1,3,4,5,7)	6	60
	(0,1,3,4,6,7)	6	121
	(0,2,3,4,5,7)	6	31
	(0,1,2,3,4,5,7)	7	7
	(0,1,2,3,4,6,7)	7	21
	(0,1,2,3,5,6,7)	7	35

where $B = (b_0(\alpha), b_1(\alpha), \cdots, b_{p+r-1}(\alpha))$ is the received array, and all the operations are modulo $M_p(x)$. As in (14), the decoding algorithms presented in the sequel heavily rely on the operation of testing whether for some $0 \le k \le p-1$ we have $a(\alpha) = \alpha^k b(\alpha)$. If such a relation holds we say that the polynomials $a(\alpha)$ and $b(\alpha)$ are *cyclically equivalent* and denote $a(\alpha) \equiv b(\alpha)$. An efficient technique for establishing cyclic equivalence may be found in Shiloach [15]. Note that, in view of Lemma 2.1, if $a(\alpha) \equiv b(\alpha)$ there exists a *unique* integer k, such that $a(\alpha) = \alpha^k b(\alpha)$ and $0 \le k \le p-1$.

We now describe a procedure for correcting up to two symbol errors using the codes $\mathcal{A}(p,r)$ for $r \ge 4$. Indeed, we assume throughout that p and r are such that the minimum distance of $\mathcal{A}(p,r)$ is ≥ 5. Since $\mathcal{A}(p,r)$ is systematic, it would suffice to correct only those errors that occurred in the information part. Let $s_0(\alpha), s_1(\alpha), s_2(\alpha), s_3(\alpha)$ be the syndrome values as defined in (15). The following algorithm finds the error locations, provided no more than two errors have occured.

Algorithm 3.1:

1) If at least two of the syndromes $s_0(\alpha), s_1(\alpha), s_2(\alpha), s_3(\alpha)$ are zero, declare no errors in the information and stop. Otherwise initialize $l \leftarrow -1$.
2) Set $l \leftarrow l+1$. If $l = p$ declare that more than two errors occurred and stop.
3) Compute:

$$\begin{aligned} y_1(\alpha) &= s_1(\alpha) + \alpha^l s_0(\alpha) \\ y_2(\alpha) &= s_2(\alpha) + \alpha^l s_1(\alpha) \\ y_3(\alpha) &= s_3(\alpha) + \alpha^l s_2(\alpha) \\ y_4(\alpha) &= s_3(\alpha) + \alpha^{3l} s_0(\alpha). \end{aligned} \tag{16}$$

4) If at least two of the polynomials $y_i(\alpha)$ and $y_j(\alpha)$ in (16) are zero and these polynomials are consecutive, namely $j = i+1 \pmod 4$, declare a single error in the information, at position l, and stop.
5) If $y_2(\alpha) \not\equiv y_1(\alpha)$ goto 2). Otherwise, let k be such that $y_2(\alpha) = \alpha^k y_1(\alpha)$.
6) If $y_3(\alpha) \not\equiv \alpha^k y_2(\alpha)$ goto 2). Otherwise, declare errors at positions l and k, and stop.

Note that multiplication by α modulo $M_p(x)$ is essentially a cyclic rotation, possibly followed by complementing all the bits of the result. We will refer to such operation as rotation modulo $M_p(x)$. Thus in the worst case the computational complexity of Algorithm 3.1 is $5p$ rotations modulo $M_p(x)$, $9p+4$ vector additions and/or comparisons, and p tests for cyclic equivalence using [15]. Furthermore, the computation of the syndromes $s_0(\alpha), s_1(\alpha), s_2(\alpha), s_3(\alpha)$ requires about $4p$ rotations modulo $M_p(x)$ and vector additions.

We presently show the correctness of Algorithm 3.1. In the following we assume that τ symbol errors have occurred.

Proposition 3.1: Algorithm 4.1 produces the true error locations, provided $\tau \le 2$.

Proof: Assuming $\tau \le 2$, it is obvious that there are no errors in the information part if and only if at least two of the syndromes are zero. Thus step 1) of the algorithm is correct, and we need to distinguish between three cases.

Case 1: A single symbol error $e_1(\alpha)$ at position i_1 in the information part. The syndrome values are given by

$$\begin{aligned} s_0(\alpha) &= e_1(\alpha) \\ s_1(\alpha) &= \alpha^{i_1} e_1(\alpha) \\ s_2(\alpha) &= \alpha^{2i_1} e_1(\alpha) \\ s_3(\alpha) &= \alpha^{3i_1} e_1(\alpha). \end{aligned} \tag{17}$$

Case 2: A single symbol error $e_1(\alpha)$ at position i_1 in the information part and a single error in the redundancy part. Here some three syndrome values are given by (17) and the other one is arbitrary.

Case 3: Two symbol errors $e_1(\alpha)$ and $e_2(\alpha)$ at positions i_1 and i_2 in the information part. The syndrome values are

$$\begin{aligned} s_0(\alpha) &= e_1(\alpha) + e_2(\alpha) \\ s_1(\alpha) &= \alpha^{i_1} e_1(\alpha) + \alpha^{i_2} e_2(\alpha) \\ s_2(\alpha) &= \alpha^{2i_1} e_1(\alpha) + \alpha^{2i_2} e_2(\alpha) \\ s_3(\alpha) &= \alpha^{3i_1} e_1(\alpha) + \alpha^{3i_2} e_2(\alpha). \end{aligned}$$

Let us consider Case 3 first. In this case, for $l = i_1$ we have in step 3) of the algorithm

$$\begin{aligned} y_1(\alpha) &= \alpha^{i_1}e_1(\alpha) + \alpha^{i_2}e_2(\alpha) + \alpha^{i_1}e_1(\alpha) + \alpha^{i_1}e_2(\alpha) \\ &= \alpha^{i_2}e_2(\alpha) + \alpha^{i_1}e_2(\alpha) \\ y_2(\alpha) &= \alpha^{2i_1}e_1(\alpha) + \alpha^{2i_2}e_2(\alpha) + \alpha^{i_1+i_1}e_1(\alpha) \\ &\quad + \alpha^{i_2+i_1}e_2(\alpha) \\ &= \alpha^{2i_2}e_2(\alpha) + \alpha^{i_2+i_1}e_2(\alpha) \\ y_3(\alpha) &= \alpha^{3i_1}e_1(\alpha) + \alpha^{3i_2}e_2(\alpha) + \alpha^{2i_1+i_1}e_1(\alpha) \\ &\quad + \alpha^{2i_2+i_1}e_2(\alpha) \\ &= \alpha^{3i_2}e_2(\alpha) + \alpha^{2i_2+i_1}e_2(\alpha) \end{aligned}$$

Hence, for $k = i_2$

$$\begin{aligned} \alpha^k y_1(\alpha) &= \alpha^{2i_2}e_2(\alpha) + \alpha^{i_2+i_1}e_2(\alpha) &= y_2(\alpha) \\ \alpha^k y_2(\alpha) &= \alpha^{3i_2}e_2(\alpha) + \alpha^{2i_2+i_1}e_2(\alpha) &= y_3(\alpha) \end{aligned}$$

Thus Algorithm 3.1 would correctly declare errors at locations $l = i_1$ and $k = i_2$. It remains to be verified that no other values of $l, k \neq i_1, i_2$ can be obtained as output of the algorithm. Assume to the contrary that there are two such values l and k. Define $\tilde{e}_2(\alpha)$ by the equation $y_1(\alpha) = \alpha^l\tilde{e}_2(\alpha) + \alpha^k\tilde{e}_2(\alpha)$, or equivalently

$$\tilde{e}_2(\alpha) = \frac{s_1(\alpha) + \alpha^l s_0(\alpha)}{\alpha^l + \alpha^k}.$$

Note that $\tilde{e}_2(\alpha)$ is well-defined since $(\alpha^l + \alpha^k)$ is invertible mod $M_p(x)$ in view of Lemma 2.1. Further define $\tilde{e}_1(\alpha) = s_0(\alpha) - \tilde{e}_2(\alpha)$. Using the relations between $y_1(\alpha), y_2(\alpha), y_3(\alpha)$ and the syndromes, along with the fact that

$$\begin{aligned} y_1(\alpha) &= \alpha^l\tilde{e}_2(\alpha) + \alpha^k\tilde{e}_2(\alpha) \\ y_2(\alpha) &= \alpha^k y_1(\alpha) \\ y_3(\alpha) &= \alpha^k y_2(\alpha) \end{aligned}$$

it may be easily verified that the errors $\tilde{e}_1(\alpha)$ and $\tilde{e}_2(\alpha)$ at positions l and k produce the same syndrome values as the errors $e_1(\alpha)$ and $e_2(\alpha)$ at positions i_1 and i_2. This clearly contradicts the fact that the minimum Hamming distance of $\mathcal{A}(p,r)$ is ≥ 5. Hence, the algorithm decodes correctly if we are in Case 3. Finally, if we are in Case 1 then for $l = i_1$ we have $y_1(\alpha) = y_2(\alpha) = y_3(\alpha) = y_4(\alpha) = 0$, while if we are in Case 2 then for $l = i_1$ at least two consecutive polynomials in (16) are zero. Furthermore, if some two consecutive polynomials in (16) are zero for some l, we can always construct an error pattern consisting of a single error in position l in the information, and possibly a single error in the redundancy, which produces the syndrome values $s_0(\alpha), s_1(\alpha), s_2(\alpha), s_3(\alpha)$. An argument similar to the above can be now employed to show that Algorithm 3.1 correctly terminates at step 4) with $l = i_1$. □

Once the error locations have been found using Algorithm 3.1, the error values may be determined as follows. In case of a single error at position l, the error value is given by $e_1(\alpha) = \alpha^{-lh}s_h(\alpha)$, where $s_h(\alpha)$ is any "redundancy-error free" syndrome. For instance, if $y_i(\alpha) = y_j(\alpha) = 0$ are the two consecutive polynomials in step 4) of Algorithm 3.1, we may always take $h = j-1$. In case of two errors at positions l and k, the error values are

$$e_2(\alpha) = \frac{y_1(\alpha)}{\alpha^l + \alpha^k} \qquad e_1(\alpha) = s_0(\alpha) + e_2(\alpha). \quad (18)$$

Notice that the inverse of $(\alpha^l + \alpha^k)$ exists by Lemma 2.1, and may be computed as follows.

Lemma 3.2 (cf. [4]): Let

$$a(\alpha) = \sum_{i=0}^{p-2} a_i\alpha^i.$$

For any $k \not\equiv l \pmod p$, the coefficients of the unique solution

$$b(\alpha) = \sum_{i=0}^{p-2} b_i\alpha^i$$

for

$$b(\alpha) = \frac{a(\alpha)}{\alpha^l + \alpha^k}$$

are given by[1]

$$b_{\langle 2j(l-k)-1\rangle_p} = \bigoplus_{i=0}^{2j-1} a_{\langle i(l-k)+l-1\rangle_p}$$

for $j = 1, 2, \cdots, p-1$, and $b_{p-1} = 0$.

Example 3.1: Consider the code $\mathcal{A}(5,4)$. Assume that the all-zero is the original codeword, and that the error-corrupted array is given by:

$$B = \begin{array}{|c|c|c|c|c|c|c|c|c|} \hline 0&0&0&1&0&0&0&0&0 \\ \hline 0&1&0&1&0&0&0&0&0 \\ \hline 0&1&0&0&0&0&0&0&0 \\ \hline 0&0&0&1&0&0&0&0&0 \\ \hline \end{array}$$

Calculating the syndromes as in (15) we obtain

$$\begin{aligned} s_0(\alpha) &= 1 + \alpha^2 + \alpha^3 &= (1011) \\ s_1(\alpha) &= 1 + \alpha^3 &= (1001) \\ s_2(\alpha) &= \alpha + \alpha^2 + \alpha^3 &= (0111) \\ s_3(\alpha) &= \alpha^2 &= (0010). \end{aligned}$$

Algorithm 3.1 starts with $l \leftarrow 0$ at step 3). In this case we have

$$\begin{aligned} y_1(\alpha) &= s_1(\alpha) + s_0(\alpha) &= (0010) \\ y_2(\alpha) &= s_2(\alpha) + s_1(\alpha) &= (1110) \\ y_3(\alpha) &= s_3(\alpha) + s_2(\alpha) &= (0101) \\ y_4(\alpha) &= s_3(\alpha) + s_0(\alpha) &= (1001). \end{aligned}$$

Clearly $y_2(\alpha) \not\equiv y_1(\alpha)$, so the algorithm goes back to step 2) where $l \leftarrow 1$. For $l = 1$ we have

$$\begin{aligned} y_1(\alpha) &= s_1(\alpha) + \alpha s_0(\alpha) &= (0011) \\ y_2(\alpha) &= s_2(\alpha) + \alpha s_1(\alpha) &= (1100) \\ y_3(\alpha) &= s_3(\alpha) + \alpha s_2(\alpha) &= (1110) \\ y_4(\alpha) &= s_3(\alpha) + \alpha^3 s_0(\alpha) &= (1100). \end{aligned}$$

Now $y_2(\alpha) = \alpha^3 y_1(\alpha)$ and $y_3(\alpha) = \alpha^3 y_2(\alpha)$. Hence the algorithm correctly declares that the errors occurred in positions 1 and 3. These errors are corrected using (18) and Lemma 3.2.

[1]The notation $\langle x\rangle_p$ is used throughout this paper to denote the unique integer x', such that $x' \equiv x \pmod p$ and $0 \leq x' \leq p-1$.

TABLE IV
ALGORITHM 4.2—SINGLE ERROR IN THE INFORMATION.
(Here $e(\alpha)$ and $\varepsilon(\alpha)$ are arbitrary values of the errors in the redundancy part.)

$z_1(\alpha)$	$z_2(\alpha)$	$z_3(\alpha)$	$z_4(\alpha)$	$z_5(\alpha)$	Error positions
0	0	0	0	0	l
$\alpha^l e(\alpha)$	0	0	0	0	l,p
$e(\alpha)$	$\alpha^l e(\alpha)$	0	0	0	$l,p+1$
0	$e(\alpha)$	$\alpha^l e(\alpha)$	0	0	$l,p+2$
0	0	$e(\alpha)$	$\alpha^l e(\alpha)$	0	$l,p+3$
0	0	0	$e(\alpha)$	$\alpha^l e(\alpha)$	$l,p+4$
0	0	0	0	$e(\alpha)$	$l,p+5$
$\alpha^l e(\alpha)+\varepsilon(\alpha)$	$\alpha^l \varepsilon(\alpha)$	0	0	0	$l,p,p+1$
$\alpha^l e(\alpha)$	$\varepsilon(\alpha)$	$\alpha^l \varepsilon(\alpha)$	0	0	$l,p,p+2$
$\alpha^l e(\alpha)$	0	$\varepsilon(\alpha)$	$\alpha^l \varepsilon(\alpha)$	0	$l,p,p+3$
$\alpha^l e(\alpha)$	0	0	$\varepsilon(\alpha)$	$\alpha^l \varepsilon(\alpha)$	$l,p,p+4$
$\alpha^l e(\alpha)$	0	0	0	$\varepsilon(\alpha)$	$l,p,p+5$
$e(\alpha)$	$\alpha^l e(\alpha)+\varepsilon(\alpha)$	$\alpha^l \varepsilon(\alpha)$	0	0	$l,p+1,p+2$
$e(\alpha)$	$\alpha^l e(\alpha)$	$\varepsilon(\alpha)$	$\alpha^l \varepsilon(\alpha)$	0	$l,p+1,p+3$
$e(\alpha)$	$\alpha^l e(\alpha)$	0	$\varepsilon(\alpha)$	$\alpha^l \varepsilon(\alpha)$	$l,p+1,p+4$
$e(\alpha)$	$\alpha^l e(\alpha)$	0	0	$\varepsilon(\alpha)$	$l,p+1,p+5$
0	$e(\alpha)$	$\alpha^l e(\alpha)+\varepsilon(\alpha)$	$\alpha^l \varepsilon(\alpha)$	0	$l,p+2,p+3$
0	$e(\alpha)$	$\alpha^l e(\alpha)$	$\varepsilon(\alpha)$	$\alpha^l \varepsilon(\alpha)$	$l,p+2,p+4$
0	$e(\alpha)$	$\alpha^l e(\alpha)$	0	$\varepsilon(\alpha)$	$l,p+1,p+5$
0	0	$e(\alpha)$	$\alpha^l e(\alpha)+\varepsilon(\alpha)$	$\alpha^l \varepsilon(\alpha)$	$l,p+3,p+4$
0	0	$e(\alpha)$	$\alpha^l e(\alpha)$	$\varepsilon(\alpha)$	$l,p+3,p+5$
0	0	0	$e(\alpha)$	$\alpha^l e(\alpha)+\varepsilon(\alpha)$	$l,p+4,p+5$

TABLE V
ALGORITHM 4.2—TWO ERRORS IN THE INFORMATION.
(Here $e(\alpha)$ is an arbitrary value of the error in the redundancy part.)

$y_1(\alpha)$	$y_2(\alpha)$	$y_3(\alpha)$	$y_4(\alpha)$	Error positions
0	0	0	0	j,l
$\alpha^{j+l}e(\alpha)$	0	0	0	j,l,p
$(\alpha^j+\alpha^l)e(\alpha)$	$\alpha^{j+l}e(\alpha)$	0	0	$j,l,p+1$
$e(\alpha)$	$(\alpha^j+\alpha^l)e(\alpha)$	$\alpha^{j+l}e(\alpha)$	0	$j,l,p+2$
0	$e(\alpha)$	$(\alpha^j+\alpha^l)e(\alpha)$	$\alpha^{j+l}e(\alpha)$	$j,l,p+3$
0	0	$e(\alpha)$	$(\alpha^j+\alpha^l)e(\alpha)$	$j,l,p+4$
0	0	0	$e(\alpha)$	$j,l,p+5$

B. Correcting Three Symbols in Error

The algorithm for correcting up to $\tau = 3$ symbol errors with $\mathcal{A}(p,r)$ for $r \geq 6$ is in principle similar to Algorithm 3.1. As before, let $s_j(\alpha)$ for $j = 0,1,\cdots,5$ be the syndromes with respect to $H(p,r)$, computed as in (15). The following algorithm produces the error locations.

Algorithm 3.2:

1) If at least three of the syndromes $s_0(\alpha), s_1(\alpha), s_2(\alpha), s_3(\alpha), s_4(\alpha), s_5(\alpha)$ are zero, declare no errors in the information and stop. Otherwise, initialize $l \leftarrow -1$.

2) Set $l \leftarrow l+1$. If $l = p$ declare that more than three errors occurred and stop. Otherwise, compute

$$\begin{aligned} z_1(\alpha) &= s_1(\alpha)+\alpha^l s_0(\alpha) \\ z_2(\alpha) &= s_2(\alpha)+\alpha^l s_1(\alpha) \\ z_3(\alpha) &= s_3(\alpha)+\alpha^l s_2(\alpha) \\ z_4(\alpha) &= s_4(\alpha)+\alpha^l s_3(\alpha) \\ z_5(\alpha) &= s_5(\alpha)+\alpha^l s_4(\alpha) \end{aligned}$$

3) If the polynomials $z_1(\alpha), z_2(\alpha), z_3(\alpha), z_4(\alpha), z_5(\alpha)$ are as given in Table IV declare a single error in the information, at position l, and stop. Otherwise, initialize $j \leftarrow -1$.

4) Set $j \leftarrow j+1$. If $j = l$ goto 2). Otherwise, compute

$$\begin{aligned} y_1(\alpha) &= z_2(\alpha) + \alpha^j z_1(\alpha) \\ y_2(\alpha) &= z_3(\alpha) + \alpha^j z_2(\alpha) \\ y_3(\alpha) &= z_4(\alpha) + \alpha^j z_3(\alpha) \\ y_4(\alpha) &= z_5(\alpha) + \alpha^j z_4(\alpha) \end{aligned}$$

5) If the polynomials $y_1(\alpha), y_2(\alpha), y_3(\alpha), y_4(\alpha)$ are as given in Table V declare two errors in the information, at positions j and l, and stop.
6) If $y_2(\alpha) \not\equiv y_1(\alpha)$ goto 4). Otherwise, let k be such that $y_2 = \alpha^k y_1(\alpha)$.
7) If either $y_3(\alpha) \neq \alpha^k y_2(\alpha)$ or $y_4(\alpha) \neq \alpha^k y_3(\alpha)$ goto 4). Otherwise, declare three errors at positions j, k, and l, and stop.

Note that the table lookup at steps 3) and 5) of the algorithm may be avoided at the expense of some additional computations. Consider for instance step 3). Clearly, there is a single error in the information if and only if some four out of the six syndromes $s_0(\alpha), s_1(\alpha), s_2(\alpha), s_3(\alpha), s_4(\alpha), s_5(\alpha)$ are appropriate rotations of each other mod $M_p(x)$, while the other two are arbitrary. Comparing a polynomial

$$z(\alpha) = s_j(\alpha) + \alpha^k s_i(\alpha)$$

to zero is equivalent to testing whether $s_j(\alpha)$ is a kth rotation of $s_i(\alpha)$ modulo $M_p(x)$. Hence, instead of the table lookup in step 3) we could proceed as follows. Compute

$$\begin{aligned} z_1(\alpha) &= s_1(\alpha) + \alpha^l s_0(\alpha) \\ z_2(\alpha) &= s_2(\alpha) + \alpha^l s_1(\alpha) \\ z_3(\alpha) &= s_3(\alpha) + \alpha^l s_2(\alpha) \\ z_4(\alpha) &= s_4(\alpha) + \alpha^l s_3(\alpha) \\ z_5(\alpha) &= s_5(\alpha) + \alpha^l s_4(\alpha) \\ z_6(\alpha) &= s_5(\alpha) + \alpha^{5l} s_0(\alpha). \end{aligned} \tag{19}$$

If at least five of the polynomials in (19) are zero, declare a single error at position l and stop. Otherwise, if at least three of the polynomials in (19) are zero, compute

$$\begin{aligned} z_7(\alpha) &= s_3(\alpha) + \alpha^{3l} s_0(\alpha) \\ z_8(\alpha) &= s_4(\alpha) + \alpha^{3l} s_1(\alpha) \\ z_9(\alpha) &= s_5(\alpha) + \alpha^{3l} s_2(\alpha). \end{aligned} \tag{20}$$

If at least one of the polynomials in (20) is zero declare a single error at position l and stop. Otherwise, if exactly two polynomials $z_i(\alpha)$ and $z_j(\alpha)$ in (19) are zero and $j - i \equiv 1, 3 \pmod 6$ then compute $z_k(\alpha)$, where $k = 7 + \langle i \rangle_3$, according to (20). If $z_k(\alpha) = 0$ declare a single error at position l and stop. Otherwise, initialize $j \leftarrow -1$ and go to step 4).

The table lookup at step 5) of Algorithm 3.2 could be rendered into computations in a similar fashion.

The computations in (19) and (20) are best explained in terms of a graph in Fig. 1, where vertices are the six syndromes and edges labeled z_i are either present or not according as $z_i(\alpha) = 0$ or $z_i(\alpha) \neq 0$ in (19) and (20). Clearly, there is a single error at position l if and only if the resulting graph contains a connected subgraph with four vertices.

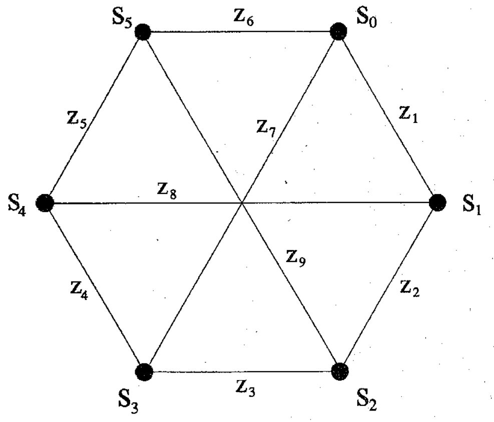

Fig. 1. Relations between the syndromes in Algorithm 4.2.

Proposition 3.3: Algorithm 4.2 produces the true error locations, provided at most three symbol errors have occurred.

The proof of Proposition 3.3 involves considering the various cases of errors in the information part and in the redundancy part. It is, in principle, similar to the proof of Proposition 3.1 and is therefore omitted.

Note that most computations in Algorithm 3.2 are of the same type. In fact, Algorithm 3.2, as well as Algorithm 3.1, may be carried out using only three kinds of operations

- testing for cyclic equivalence: $a(\alpha) \overset{?}{\equiv} b(\alpha)$
- rotate and add: $c(\alpha) \leftarrow a(\alpha) + \alpha^k b(\alpha)$
- comparison with zero: $a(\alpha) \overset{?}{=} \underline{0}$.

Each of these operations is readily implementable in special-purpose hardware. The worst case complexity of Algorithm 3.2, not including the syndrome computation and the table lookup, is given by $\frac{1}{2}p(p-1)$ tests for cyclic equivalence, $p(3p+2)$ rotate-and-add operations, and $p(p-1)+5$ comparisons with $\underline{0}$. In the same terms, the worst case complexity of Algorithm 3.1 is p tests for cyclic equivalence, $5p$ rotate-and-add operations, and $5p+4$ comparisons with $\underline{0}$.

Once the error locations have been computed, the error values may be reconstructed as follows. In case of a single error at position l, we have $e_1(\alpha) = \alpha^{-lh} s_h(\alpha)$, where $s_h(\alpha)$ is any of the four connected vertices in Fig. 1. In case of two errors at positions j and l, we have

$$\begin{aligned} e_1(\alpha) &= \frac{\alpha^{-(i-1)j} z_i(\alpha)}{\alpha^j + \alpha^l} \\ e_2(\alpha) &= \alpha^{-il}\left(s_i(\alpha) + \alpha^{ij} e_2(\alpha)\right) \end{aligned} \tag{21}$$

where $i \in \{1,2,3,4\}$ is such that $y_i(\alpha) = 0$, and the division in (21) is computed as in Lemma 3.2. In case of three errors at positions j, k, l, we employ the results of [4] to arrive at

$$\begin{aligned} e_1(\alpha) &= \frac{y_1(\alpha)}{(\alpha^k + \alpha^j)(\alpha^k + \alpha^l)} \\ e_2(\alpha) &= \frac{z_1(\alpha) + (\alpha^k + \alpha^l) e_1(\alpha)}{\alpha^j + \alpha^l} \end{aligned}$$

and

$$e_3(\alpha) = s_0(\alpha) + e_1(\alpha) + e_2(\alpha).$$

This completes the error-correction procedure.

Finally, it is worth mentioning that Algorithm 3.2 in effect amounts to p successive invocations of a slightly modified version of Algorithm 3.1 with $z_1(\alpha), z_2(\alpha), \cdots, z_5(\alpha)$ substituted for the syndrome values. This provides a way to extend our approach to the correction of an arbitrary number of errors. In order to correct τ symbol errors with $\mathcal{A}(p,r)$ for $r \geq 2\tau$, compute for each $l = 0, 1, \cdots, p-1$

$$\begin{aligned} z_1(\alpha) &= s_1(\alpha) + \alpha^l s_0(\alpha) \\ z_2(\alpha) &= s_2(\alpha) + \alpha^l s_1(\alpha) \\ &\vdots \\ z_{2\tau-1}(\alpha) &= s_{2\tau-1}(\alpha) + \alpha^l s_{2\tau-2}(\alpha) \end{aligned}$$

and substitute $z_1(\alpha), z_2(\alpha), \cdots, z_{2\tau-1}(\alpha)$ for the syndrome values into an appropriately modified version of the decoder for $\mathcal{A}(p, r-2)$. However, the complexity of such a decoding scheme is obviously $O(p^{\tau-1})$. At present we do not have a technique which would allow to avoid exponential complexity *and* finite field operations at the same time. Nevertheless, for the small values of $\tau = 2$ and $\tau = 3$ the proposed decoding algorithms are quite feasible.

IV. Optimality of the Updates

In this section we address the issue of updating the parity (or redundancy) symbols following an update in the information. Let C be a systematic linear (n,k) code over GF(2^m), and assume that the symbols of C are represented as binary m-tuples. We define $\eta(C)$ to be the average number of parity bits affected by a change in a single information bit in C. More precisely, for any $\underline{c} = (c_0, c_1, \cdots, c_{n-1}) \in C$ let wt(c_i) denote the Hamming weight of the ith symbol in $\underline{c}$ regarded as a binary m-tuple, and let

$$\mathcal{S} = \{ (c_0, c_1, \cdots, c_{n-1}) \in C : \sum_{i=0}^{k-1} \text{wt}(c_i) = 1 \} \quad (22)$$

be a subset of C consisting of all the codewords that contain a single nonzero entry in the information part. Then

$$\eta(C) \overset{\text{def}}{=} \frac{1}{mk} \sum_{\underline{c}\in\mathcal{S}} \sum_{i=k}^{n-1} \text{wt}(c_i).$$

This parameter $\eta(C)$ is of crucial importance in storage applications that require frequent updates of information. Indeed, in such applications it is desirable to use codes for which $\eta(C)$ is as small as possible. This is precisely the property **P3** mentioned in the Introduction.

Herein we prove upper and lower bounds on $\eta(C)$ for MDS codes over GF(2^m). In particular, we show that if $C = \mathcal{A}(p,r)$, with symbols of size $p-1$, then $\eta(C)$ is upper-bounded by $2r-1$. It follows, therefore, that for our codes $\eta(C)$ *does not depend on the size of the symbols.* In contrast, it is shown that for Reed–Solomon codes, as well as for the MDS codes of Blaum and Roth [4], $\eta(C)$ increases linearly with the symbol size. Thus our codes are indeed more suitable for use with very large symbols.

Proposition 4.1: For $C = \mathcal{A}(p,r)$, we have

$$\eta(C) = 2r - 1 - \frac{2(r-1)}{p}.$$

Proof: Consider column 0 in any codeword of $\mathcal{A}(p,r)$. If any one of the $p-1$ bits in this column is changed, we need to make exactly r updates—one in each parity symbol. Therefore, column 0 requires a total of $(p-1)r$ updates. Now consider column l, where $1 \leq l \leq p-1$. For each of the $p-1$ information bits in this column we shall count the number of parity bits that are affected by a change in that bit. Consider first the $r-1$ information bits in entries $(\langle -il-1\rangle_p, l)$ for $i = 1, 2, \cdots, r-1$. Since p is prime we have $\langle -il-1\rangle_p \neq \langle -jl-1\rangle_p$ for all $1 \leq i < j \leq r-1$, which implies that all these entries are distinct. When entry $(\langle -il-1\rangle_p, l)$ is changed, one has to to update all the $p-1$ bits in parity column $p+i$, as well as one bit in each of the parity columns $p+j$ for $j = 0, 1, \cdots, i-1, i+1, \cdots, r-1$. This gives a total of $(r-1)(p+r-2)$ updates for these $r-1$ information bits. The remaining $p-r$ bits in the lth column require r updates each. Thus the total number of updates for all the bits in column l is $(r-1)(p+r-2)+r(p-r)$. Since $l \neq 0$, but otherwise arbitrary, we may now compute

$$\begin{aligned} \eta(C) &= \frac{(p-1)r + \sum_{l=1}^{p-1} [(r-1)(p+r-2) + r(p-r)]}{p(p-1)} \\ &= 2r - 1 - \frac{2(r-1)}{p}. \end{aligned} \qquad \square$$

For instance, if $r = 1$ then $\eta(C) = 1$ which corresponds to the trivial parity code, and is obviously optimal. For $C = \mathcal{A}(p,2)$ we have $\eta(C) = 3 - 2/p \approx 3$. In general, for $\mathcal{A}(p,r)$ the average number of updates can be approximated by $\eta(C) \approx 2r-1$, which indeed does not depend on the size of the symbols.

However, is $2r-1$ optimal for MDS codes with the parameters of $\mathcal{A}(p,r)$? A trivial lower bound for an MDS code C with minimum distance $r+1$ is $\eta(C) \geq r$, since a change in any information bit must affect all r parity symbols. It is an interesting open problem to narrow the gap between the upper and lower bounds $r \leq \eta(C) \leq 2r-1$ for MDS codes with the parameters of $\mathcal{A}(p,r)$. We presently show that the lower bound is unattainable for $r \geq 2$.

Proposition 4.2: Let C be an MDS code with symbols of size $p-1$ bits, having at least p information symbols and r parity symbols, where $r \geq 2$. Then $\eta(C) > r$.

Proof: We will assume that $\eta(C) = r$ and reach a contradiction. W.l.o.g. suppose that the number of information symbols is p, and let $\mathcal{S}$ be the subset of C as defined in (22). Clearly, $|\mathcal{S}| = p(p-1)$. It is also clear that for every codeword in $\mathcal{S}$ none of the r parity symbols is zero, since otherwise the minimum distance of C would be $\leq r$. Moreover, since we assume that $\eta(C) = r$, each parity symbol in every codeword of $\mathcal{S}$ has weight exactly 1. Hence, there are at most $(p-1)^2$ distinct ways to choose the first two parity symbols in a codeword of $\mathcal{S}$. Since $|\mathcal{S}| = p(p-1) > (p-1)^2$ it follows that there are some two codewords $\underline{u}, \underline{v} \in \mathcal{S}$ which coincide in the first two parity symbols. But then the Hamming distance between $\underline{u}$ and $\underline{v}$ is at most r, contradicting the assumption that the minimum distance of C is $r+1$. □

The proof of Proposition 4.2 is based on the fact that the number of information symbols is sufficiently large relative to the size of the symbols, and applies to both linear and nonlinear codes. Next we show that for linear MDS codes we cannot have $\eta(C) = r$ even if there are only two information symbols. This claim is slightly stronger than that of Proposition 4.2.

Proposition 4.3: Let C be a linear MDS code having at least two information symbols and r parity symbols, where $r \geq 2$. Then $\eta(C) > r$.

Proof: Again, we assume that $\eta(C) = r$ and reach a contradiction. Let the first two symbols in each codeword of C be information symbols, and define

$$\begin{aligned} \mathcal{S}_0 &= \{(c_0, c_1, \cdots, c_{n-1}) \in C : \mathrm{wt}(c_0) = 1 \text{ and } \mathrm{wt}(c_i) = 0 \\ &\qquad \text{for } i = 1, 2, \cdots, n-r-1\} \\ \mathcal{S}_1 &= \{(c_0, c_1, \cdots, c_{n-1}) \in C : \mathrm{wt}(c_1) = 1 \text{ and } \mathrm{wt}(c_i) = 0 \\ &\qquad \text{for } i = 0, 2, 3, \cdots, n-r-1\} \end{aligned}$$

If $(c_0, c_1, \cdots, c_{n-1})$ and $(c_0', c_1', \cdots, c_{n-1}')$ are any two codewords in $\mathcal{S}_0$ then clearly

$$c_{n-r}' \neq c_{n-r}, c_{n-r+1}' \neq c_{n-r+1}, \cdots, c_{n-1}' \neq c_{n-1}$$

since the minimum distance of $\mathcal{S}_0$ is $r+1$. Furthermore, by the assumption $\eta(C) = r$ it follows that

$$\mathrm{wt}(c_{n-r}) = \mathrm{wt}(c_{n-r+1}) = \cdots = \mathrm{wt}(c_{n-1}) = 1$$

for any $(c_0, c_1, \cdots, c_{n-1}) \in \mathcal{S}_0$. Hence

$$\underline{s}_0 = \sum_{\underline{c} \in \mathcal{S}_0} \underline{c} = (\underbrace{\underline{1}, \underline{0}, \underline{0}, \cdots, \underline{0}}_{n-r}, \underbrace{\underline{1}, \underline{1}, \cdots, \underline{1}}_{r})$$

where $\underline{0}, \underline{1}$ denote the all-zero and all-one binary columns, respectively. By a similar argument

$$\underline{s}_1 = \sum_{\underline{c} \in \mathcal{S}_1} \underline{c} = (\underbrace{\underline{0}, \underline{1}, \underline{0}, \underline{0}, \cdots, \underline{0}}_{n-r}, \underbrace{\underline{1}, \underline{1}, \cdots, \underline{1}}_{r}).$$

But then by linearity of C we have

$$\underline{s}_0 + \underline{s}_1 = (\underline{1}, \underline{1}, \underline{0}, \underline{0}, \ldots, \underline{0}) \in C$$

which contradicts the fact that the minimum distance of C is $r + 1 \geq 3$. □

We note here that, although Proposition 4.3 is stronger than Proposition 4.2, the proof technique of Proposition 4.2 can be used to show that in any sufficiently long MDS code $\eta(C)$ must increase linearly with the symbol size. Consider, for instance, Reed–Solomon codes over GF(2^m) with $O(2^m)$ information symbols. Then the size of the set $\mathcal{S} \subset C$ as defined in (22) is $O(m2^m)$. Now assume that $\eta(C) \leq \lambda$ where λ does not depend on m. Then the number of ways to choose the first two parity symbols in a codeword of $\mathcal{S}$ is upper-bounded by $O(m^{2\lambda})$, which is less than $O(m2^m)$ for any constant λ and sufficiently large m. Hence, as in the proof of Proposition 4.2, we arrive at a contradiction with the minimum distance property of the code. This implies that in Reed–Solomon codes with $O(2^m)$ information symbols $\eta(C)$ must increase with the symbol size. A more careful counting argument along the same lines shows that in fact $\eta(C)$ increases *linearly* with the size of the symbols. This follows essentially from the fact that for $n \to \infty$ we have

$$\binom{n}{g(n)}^2 < 2^n$$

for any sublinear function $g(n)$.

As shown above, the fact that for Reed–Solomon codes the average number of parity updates is high follows simply from the fact that these codes are long relative to the size of their symbols. However, if the number of information symbols in a code C is small relative to its symbol size, this does not necessarily mean that $\eta(C)$ will be low. Consider, for instance, the MDS array codes of Blaum and Roth [4]. These codes have symbols of size $p-1$ and contain r information symbols *less* than the codes $\mathcal{A}(p, r)$. Thus each codeword is a $(p-1) \times p$ binary array where the last r columns may be taken as the parity symbols. Assume for the time being that $r = 2$ and let $a_{p-2}(\alpha)$, $a_{p-1}(\alpha)$ denote the parity columns. It may be shown that if a single information bit in column j and row i is updated, where $0 \leq j \leq p-3$ and $0 \leq i \leq p-2$, then the parity columns must be recomputed as follows:

$$\begin{aligned} a_{p-2}(\alpha) &\leftarrow a_{p-2}(\alpha) + \alpha^{i+1} + \alpha^{i+2} + \cdots + \alpha^{i+j+1} \\ a_{p-1}(\alpha) &\leftarrow a_{p-1}(\alpha) + \alpha^{i} + \alpha^{i+1} + \cdots + \alpha^{i+j+1} \end{aligned} \qquad (23)$$

with all operations taken modulo $M_p(x)$. Using (23) and averaging over all i, j we find that $\eta(C) = \frac{2}{3}p + 1$. Thus it follows that in the Blaum–Roth codes $\eta(C)$ increases linearly with the symbol size, even though the codes of [4] are shorter than our codes. In fact, it was this shortcoming of the codes of Blaum and Roth [4] that originally motivated us to develop the codes presented in this paper. The new codes combine the advantages of the Blaum–Roth codes (MDS, encoding and decoding without finite field operations) with low $\eta(C)$.

V. Conclusions

We have presented a new family of MDS array codes whose codewords are $(p-1) \times (p+r)$ binary arrays with r independent parity columns, where $p \geq 3$ is a prime number. This family extends previously known results of [2], [4] for $r = 2$. We proved that the new codes are MDS for $r \leq 3$, and gave necessary and sufficient conditions for these codes to be MDS for $r \geq 4$. Using these conditions, we have shown that the new codes remain MDS up to $r \leq 8$ if $p > 29$ is such that 2 is primitive in GF(p). Decoding procedures which do not require finite field operations were presented for up to three symbol errors, extending the previously known results of [4] for the case of a single symbol error. Finally, we developed upper and lower bounds on the average number of parity bits affected by an update in an information bit, showing that for our codes this number does not depend on the symbol size. This property makes the new codes more suitable for use in storage applications requiring large symbols and frequent data updates, such as RAID architectures and/or holographic recording [17], than the conventional Reed–Solomon codes or the array codes of Blaum and Roth [4].

APPENDIX

This appendix contains the proof of Theorem 2.5, which gives an upper bound on the number of different classes of polynomials that must be checked to establish whether $\mathcal{A}(p,r)$ is MDS. Recall that $\mathcal{A}(p,r)$ is MDS if and only if every $r \times r$ submatrix of $H(p,r)$ is nonsingular. Any such submatrix M corresponds to a binary vector $\underline{u} = (\underline{u}' \,|\, \underline{v})$ of length $p+r$ and weight r, where the nonzero entries of $\underline{u}$ indicate the columns of $H(p,r)$ contained in M.The vector $\underline{v}$ of length r corresponds to the redundant part of the codeword, and we only need to consider the case where $\underline{v}$ is nonzero, since otherwise M is a Vandermonde matrix.

We shall write $\underline{v} = (v_0, v_1, \ldots, v_{r-1})$ and let

$$0 \le a_1 < a_2 < \cdots < a_m \le r-1$$

be the positions of zeros in $\underline{v}$, where $m = r - \mathrm{wt}(\underline{v})$. Thus we essentially deal with determinants of the type

$$D(a_1, a_2, \cdots, a_m;\, i_0, i_1, \cdots, i_{m-1}) \overset{\text{def}}{=}$$

$$\overset{\text{def}}{=} \begin{vmatrix} \alpha^{a_1 i_0} & \alpha^{a_1 i_1} & \cdots & \alpha^{a_1 i_{m-1}} \\ \alpha^{a_2 i_0} & \alpha^{a_2 i_1} & \cdots & \alpha^{a_2 i_{m-1}} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha^{a_m i_0} & \alpha^{a_m i_1} & \cdots & \alpha^{a_m i_{m-1}} \end{vmatrix} \tag{24}$$

where $0 \le i_0 < i_1 < \cdots < i_{m-1} \le p-1$ are the positions of the information columns. Determinants of the type (24) are known in the literature (cf. [11], [16]) as alternants and have been shown to be nonzero over the field of real numbers [13]. For the treatment of alternants over the field of complex numbers see [9]. However, it appears that the results of [9], [11], [13] are not applicable for polynomials modulo $M_p(x)$.

Lemma A.1: W.l.o.g. it may be assumed that $v_0 = 0$, or equivalently, that $a_1 = 0$.

Proof: Expanding M about the columns corresponding to the nonzeros in $\underline{v}$ we obtain the determinant

$$D(a_1, a_2, \cdots, a_m; i_0, i_1, \cdots, i_{m-1})$$

as defined in (24). Now

$$D(a_1, a_2, \cdots, a_m;\, i_0, \cdots, i_{m-1}) =$$
$$= \alpha^{a_1 i_0} \cdots \alpha^{a_1 i_{m-1}} D(0, a_2 - a_1, \cdots, a_m - a_1;\, i_0, \cdots, i_{m-1})$$

Hence, in view of Lemma 2.1

$$D(a_1, a_2, \cdots, a_m;\, i_0, i_1, \cdots, i_{m-1})$$

is invertible modulo $M_p(x)$ if and only if so is

$$D(0, a_2 - a_1, \cdots, a_m - a_1;\, i_0, i_1, \cdots, i_{m-1}).$$

However, this determinant corresponds to some vector $\underline{v}' = (v_0', v_1', \cdots, v_{r-1}')$ with $v_0' = 0$. □

Referring to (2), it is trivial to see that if $\mathcal{A}(p,r')$ is not MDS for some $r' < r$ then $\mathcal{A}(p,r)$ is surely not MDS. Therefore, when checking whether for a certain prime p our codes are MDS we will consider the values of r in increasing order. If for a certain value of p we find an r such that $\mathcal{A}(p,r)$ is not MDS, we may eliminate this value of p from further consideration. In this context we have the following lemma.

Lemma A.2: W.l.o.g. it may be assumed that $v_{r-1} = 0$, or equivalently, that $a_m = r-1$.

Proof: If $v_{r-1} = 1$ then expanding about the columns corresponding to the nonzeros in $\underline{v}$ produces a determinant which does not contain any entries from the last row of $H(p,r)$. Such a determinant corresponds to the case $r' < r$, and must have been already considered earlier. □

For example, the vectors $\underline{v} = (01001)$ and $\underline{v} = (00101)$ for $r = 5$ correspond to (3) and (4), respectively, already considered for $r = 4$.

Note that an argument similar to Lemma A.1 implies that

$$D(a_1, a_2, \cdots, a_m;\, i_0, i_1, \cdots, i_{m-1})$$

is invertible modulo $M_p(x)$ if and only if so is

$$D(a_1, a_2, \cdots, a_m;\, 0, k_1, \cdots, k_{m-1})$$

where $k_j = i_j - i_0$ and

$$1 \le k_1 < k_2 < \cdots < k_{m-1} \le p-1.$$

We use this fact in the following lemma, showing that the number of different cases we need to consider may be further reduced.

Lemma A.3: If the vectors $\underline{v}$ and $\underline{v}'$ are reflections of each other, that is $v_j' = v_{r-1-j}$ for all $j = 0, 1, \cdots, r-1$, then only one of these vectors need be considered.

Proof: Let $a_1', a_2', \cdots, a_m'$ denote the positions of zeros in $\underline{v}'$. Then

$$\begin{aligned} D(a_1', a_2', \cdots, a_m';\, 0, k_1, \cdots, k_{m-1}) &= \\ = \alpha^{(r-1)k_1} \alpha^{(r-1)k_2} \cdots \alpha^{(r-1)k_{m-1}} & \\ \cdot D(a_1, a_2, \cdots, a_m;\, 0, p-k_1, \cdots, p-k_{m-1}) & \end{aligned} \tag{25}$$

where we have used the fact that $a_j' = (r-1) - a_j$ for all j. Indeed, in the ring of polynomials modulo $M_p(x)$ we have

$$\begin{aligned} x^{pa} - 1 &= (x-1)M_p(x)(x^{(a-1)p} + x^{(a-2)p} + \cdots + 1) \\ &\equiv 0 \bmod M_p(x). \end{aligned}$$

Hence $\alpha^{pa_j} = 1$ and therefore

$$\alpha^{(r-1)k_l} \alpha^{a_j(p-k_l)} = \alpha^{pa_j} \alpha^{(r-1-a_j)k_l} = \alpha^{a_j' k_l}.$$

This establishes (25). Observe that when $k_1, k_2, \cdots, k_{m-1}$ range through all the possible values so do their complements $(p-k_1), (p-k_2), \cdots, (p-k_m)$. Hence

$$D(a_1, a_2, \cdots, a_m;\, 0, k_1, \cdots, k_{m-1})$$

and

$$D(a_1, a_2, \cdots, a_m;\, 0, p-k_1, \cdots, p-k_{m-1})$$

enumerate over the same polynomials. □

It follows from Lemmas A.1 and A.2 that the number of different vectors that need be checked for each r does not exceed $2^{r-2} - 1$. Lemma A.3 further eliminates half of the remaining vectors that are nonpalindromic, that is, not equal to their reflections. Since the number of nonzero binary palindromes of length $r-2$ is given by $2^{\lceil (r-2)/2 \rceil} - 1$, we conclude that Lemma A.3 eliminates precisely $(2^{r-2} - 2^{\lceil (r-2)/2 \rceil})/2$

vectors. Hence, we are left with $2^{r-3}+2^{\lceil (r-2)/2\rceil -1}-1$ cases to consider.

Finally, we can eliminate the cases in which

$$D(0, a_2, \cdots, r-1;\ 0, k_1, \cdots, k_{m-1})$$

is a Vandermonde determinant. This happens whenever a_2 divides $a_m = r-1$, and the determinant is given by

$$D(0, a_2, 2a_2, \cdots, r-1;\ 0, k_1, \cdots, k_{m-1})$$

where m stands for $(r-1)/a_2$. Therefore, for each r, we can further eliminate $\tau(r-1)$ cases, where $\tau(n)$ denotes the number of divisors of n different from 1. For instance, for $r = 7$ there are three Vandermonde determinants of this type: $D(0,2,4,6;\ 0,k_1,k_2,k_3)$, $D(0,3,6;\ 0,k_1,k_2)$, and $D(0,6;\ 0,k_1)$. They correspond to the three divisors of 6, namely 2, 3, and 6.

Combining this argument with Lemmas A.1, A.2, and A.3, we conclude that the number of different cases to consider is at most

$$\nu(r) \leq 2^{r-3}+2^{\lceil (r-2)/2\rceil -1}-\tau(r-1)-1.$$

This is precisely the upper bound on $\nu(r)$ given in Theorem 2.5.

ACKNOWLEDGMENT

The authors wish to thank M. Eberhardt for his contribution to Section II of this paper. They are also grateful to the anonymous referees for comments which helped improve the presentation of this paper. A. Vardy wishes to thank Hagit Itzkowitz for her invaluable help.

REFERENCES

[1] M. Blaum, "A class of byte-correcting array codes," IBM Res. Rep. RJ 5652 (57151), May 1987.
[2] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An optimal scheme for tolerating double disk failures in RAID architectures," *IEEE Trans. Comput.*, vol. 44, pp. 192–202, 1995.
[3] M. Blaum, H. Hao, R. Mattson, and J. Menon, "A coding technique for double disk failures in disk arrays," U.S. Patent 5 271 012, Dec. 1993.
[4] M. Blaum and R. M. Roth, "New array codes for multiple phased burst correction," *IEEE Trans. Inform. Theory*, vol. 39, pp. 66–77, 1993.
[5] T. Fuja, C. Heegard, and M. Blaum, "Cross parity check convolutional codes," *IEEE Trans. Inform. Theory*, vol. 35, pp. 1264–1276, 1989.
[6] G. Gibson, L. Hellerstein, R. M. Karp, R. H. Katz, and D. A. Patterson, "Coding techniques for handling failures in large disk arrays," Rep. UCB/CSD 88/477, Dec. 1988.
[7] R. M. Goodman, R. J. McEliece, and M. Sayano, "Phased burst error correcting array codes," *IEEE Trans. Inform. Theory*, vol. 39, pp. 684–693, 1993.
[8] R. M. Goodman and M. Sayano, "Size limits on phased burst error correcting array codes," *IEE Elect. Lett.*, vol. 26, pp. 55–56, 1990.
[9] W. Ledermann, *Introduction to Group Characters*. Cambridge, UK: Cambridge Univ. Press, 1977.
[10] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. New York: North-Holland, 1977.
[11] T. Muir, *A Treatise on the Theory of Determinants*. New York: Dover, 1960.
[12] D. A. Patterson, G. A. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks," in *SIGMOD Int. Conf. on Data Management*, (Chicago, IL, 1988), pp. 109–116.
[13] G. Pólya and G. Szegö, *Aufgaben und Lehrsatze aus der Analysis*. Berlin, Germany: Springer, 1954.
[14] P. Prusinkiewicz and S. Budkowski, "A double track error-correction code for magnetic tape," *IEEE Trans. Comput.*, vol. C-19, pp. 642–645, June 1976.
[15] Y. Shiloach, "A fast equivalence-checking algorithm for circular lists," *Inform. Proc. Lett.*, vol. 8, pp. 236–238, 1979.
[16] H. Van de Vel, "Numerical treatment of a generalized Vandermonde system of equations," *Linear Algebra Appl.*, vol. 17, pp. 149–179, 1977.
[17] A. Vardy, M. Blaum, P. H. Siegel, and G. T. Sincerbox, "Conservative arrays: multi-dimensional modulation codes for holographic recording," *IEEE Trans. Inform. Theory*, vol. 42, no. 1, pp. 227–230, Jan. 1996.