

Next we compare the dimensions of the spaces  $\mathcal{L}(G)$  and  $\mathcal{L} := \bigoplus_{\alpha \in S} \mathcal{L}(D_\alpha)g^\alpha$  for  $\deg G > 2g(F) - 2 = 26$  (this occurs iff  $3a + 6b + 2 > 0$ ). In this case,  $\dim \mathcal{L}(G) = 12a + 24b + 21$  and one can check that

$$\dim \mathcal{L} \geq \sum_{\alpha \in S} (\deg D_\alpha + 1) = 12a + 24b + 21$$

so that  $\mathcal{L}(G) = \mathcal{L}$ .

## V. CONCLUSION

By adapting Goppa's code construction, we have given a very simple construction of algebraic-geometric codes from algebraic curves which have the advantage that a basis for the code can be readily constructed. The minimum distance of these codes can be bounded below by the usual Goppa lower bound on the minimum distance and furthermore we give good upper bounds on the minimum distance of the codes. The codes constructed here are always subcodes of Goppa codes and in many cases they coincide with Goppa codes.

## REFERENCES

- [1] I. Aleshnikov, V. Deolalikar, P. V. Kumar, and H. Stichtenoth, "Toward a basis for the space of regular functions in a tower of function fields meeting the Drinfeld-Vladut bound," in *Proc. 5th Int. Conf. Finite Fields and Applications*, Augsburg, Germany, Aug. 1999.
- [2] P. Beelen and R. Pellikaan, "The Newton polygon of plane curves with many rational points," *Des., Codes Cryptogr.*, vol. 21, no. 1-3, pp. 41-67, 2000.
- [3] M. Daberkow, C. Fieker, J. Klüners, M. Pohst, K. Roegner, and K. Wildanger, "KANT V4," *J. Symbol. Comp.*, vol. 24, pp. 267-283, 1997.
- [4] D. A. Leonard, "Finding the defining functions for one-point algebraic-geometry codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 6, pp. 2566-2573, Sep. 2001.
- [5] D. A. Leonard and R. Pellikaan, "Integral closures and weight functions over finite fields," *Finite Fields Appl.*, vol. 9, no. 4, pp. 479-504, 2003.
- [6] H. Maharaj, "Code construction on fiber products of Kummer covers," *IEEE Trans. Inf. Theory*, vol. 50, no. 9, pp. 2169-2173, Sep. 2004.
- [7] —, "Explicit constructions of algebraic-geometric codes from towers," manuscript in review.
- [8] —, "Explicit Goppa Code construction on fiber products of Kummer covers using differentials," manuscript in review.
- [9] —, "Explicit constructions of algebraic geometric codes using differentials," manuscript in review.
- [10] —, "Improved lower bounds of the minimum distance of a class of Goppa codes," manuscript in review.
- [11] H. Niederreiter and C. P. Xing, "Cyclotomic function fields, Hilbert class fields, and global function fields with many rational places," *Acta Arithm.*, vol. 79, pp. 59-76, 1997.
- [12] —, *Rational Points on Curves over Finite Fields*. Cambridge, U.K.: Cambridge Univ. Press, 2001, vol. 285, London Math. Soc. Lecture Note Ser.
- [13] R. Pellikaan, H. Stichtenoth, and F. Torres, "Weierstrass semigroups in an asymptotically good tower of function fields," *Finite Fields Appl.*, vol. 4, pp. 381-392, 1998.
- [14] K. W. Shum, I. Aleshnikov, P. V. Kumar, H. Stichtenoth, and V. Deolalikar, "A low-complexity algorithm for the construction of algebraic-geometric codes better than the Gilbert-Varshamov bound," *IEEE Trans. Inf. Theory*, vol. 47, no. 6, pp. 2225-2241, Sep. 2001.
- [15] H. Stichtenoth, *Algebraic Function Fields and Codes*. Berlin, Germany: Springer-Verlag, 1993.
- [16] G. van der Geer and M. van der Vlugt, "How to construct curves over finite fields with many points," in *Arithmetic Geometry*, vol. XXXVII, Sympos. Math., Cortona, 1994, pp. 169-189.
- [17] —, "Tables of curves with many points," *Math. Comput.*, vol. 69, no. 230, pp. 797-810, 2000.
- [18] C. P. Xing, "Hyperelliptic function fields and codes," *J. Pure Appl. Alg.*, vol. 74, no. 1, pp. 109-118, 1991.

## The Burst Error Correcting Capabilities of a Simple Array Code

Dan Raphaeli, *Senior Member, IEEE*

**Abstract**—We propose a simple decoder for a widely used array code, known as the EVENODD code, which is originally designed to correct phased burst errors, to make it useful for correcting nonphased errors. The proposed scheme is capable of correcting almost all bursts up to a certain length. We show that the failure rate is sufficiently small and approaches zero as the block length increases. The redundancy of the code is twice the maximal burst length, which is a lower bound for the redundancy of a true burst-error-correcting code. Both the encoder and the decoder have very low complexity, both in terms of number of operations and in terms of computer code size.

**Index Terms**—Burst-error-correcting codes, decoding algorithms, error-correcting codes.

## I. INTRODUCTION

Burst-error-correcting codes are needed in virtually uncountable applications. Error-correcting codes designed to correct a burst of length  $l$  bits can correct any error pattern that spans not more than  $l$  bits. Such codes will be called complete burst-error-correcting codes. There are quite a few constructions for complete burst-error-correcting codes like the Fire codes and others [8]. In this correspondence, we present an error-correcting code that is not complete, since it can correct most burst patterns of length  $\leq l$  but not all of them. However, if the number of uncorrectable patterns is sufficiently small, this code can be used in practice as a burst-error-correcting code. For example, for  $l = 29$ , the probability that a burst of length  $\leq l$  is not corrected is about  $10^{-5}$ . Since the frame-error rate (FER) in communication systems would be the product of this probability and that of such a burst occurring, for most applications sufficiently low values of FER are achieved despite the limited correction capability. There are other constructions for non-complete error-correcting codes which achieve lower redundancy e.g., [11] and references therein. However, all these constructions are based on product codes and are optimized for phased errors.

The advantages of the noncomplete burst-error-correcting code presented in this correspondence are the very efficient and simple decoding algorithm, the low redundancy, and the fact that it is systematic. The redundancy of a complete burst-error-correcting code has to satisfy the Reiger bound [8]. According to this bound, the number of parity-check bits in a code should satisfy  $n - k \geq 2l$ , where  $l$  is the burst length in bits,  $k$  is the number of data bits, and  $n$  is the number of coded bits. The code presented here satisfies  $n - k = 2l$ . However, it is not a complete code. Therefore, the bound does not apply and can serve for the purpose of comparison only.

A comparison between the correction capability of the scheme proposed here and that of existing efficient complete burst-error-correcting codes with similar  $k$  and  $n$ , is presented in Table I. The first two were taken from [8, Table 9.3], and the third from [9] ( $n_1 = 31$ ,  $n_2 = 35$ ), being the best codes we have found in the literature. The multiplication

Manuscript received May 5, 2003; revised March 21, 2004. The material in this correspondence was presented in part at the IEEE Global Telecommunications Conference, Dallas, TX, November 2004.

The author is with the Faculty of Engineering, the Electrical Engineering-Systems Department, Tel-Aviv University, Tel-Aviv, Ramat-Aviv 69978, Israel (e-mail: danr@eng.tau.ac.il).

Communicated by K. A. S. Abdel-Ghaffar, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2004.840890

TABLE I  
A COMPARISON BETWEEN THE CORRECTION CAPABILITIES OF EVENODD CODES AND OPTIMAL BURST-ERROR-CORRECTING CODES

Case	Our Code	Optimal Code
1	$m=17$ , the code is (304,272), can correct 16 bits bursts	(151 × 2=302,136 × 2=272) can correct only 12 bits
2	$m=25$ , (648,600) can correct 24 bits	(217 × 3=651,202 × 3=606) can correct only 18 bits
3	$m=33$ , (1120,1056) can correct 32 bits	(1085,1020) can correct only 25 bits

by a small constant in the optimal code parameters refers to the interleaving degree applied. The parameter  $m$  used in the construction of the code will be defined shortly. It generates a code with  $k = m(m - 1)$ ,  $n = (m + 2)(m - 1)$ . It is important to note that the ability of the complete codes to decode successfully bursts longer than their nominal length deteriorates very rapidly. Simulations show that the fraction of burst patterns leading to failure in the decoding for the (151, 136) code is 0.045, 0.145, 0.23 for  $l = 7, 8, 9$ , respectively, and for the (217, 202) code the fractions are 0.032, 0.105, 0.295 for  $l = 7, 8, 9$ , respectively. Note that examining the bound developed by Gallager [13], there is a wide gap between the upper bound and the lower bound, so there is a possibility to make good incomplete cyclic codes, but it is still an open problem how to design one.

The original EVENODD scheme [2] is part of a group of codes that deal with phased burst errors [5]–[7], and can deal with a burst of errors of a specific nature that suits the original use of the method in disk backup. The scheme can handle a burst of errors that damage one disk out of an array of disks, so the errors appear in a burst and always in a known phase (the beginning of the disk).

The interest addressed by the proposed scheme is in adapting this scheme to the needs of bit-by-bit serial data communications. By grouping each sequence of  $m - 1$  bits to form one column, each disk becomes a column. According to the EVENODD scheme, a sequence of  $m$  columns is appended with two columns of parity, thus, seen as a binary code, we get  $k = m(m - 1)$ ,  $n = (m + 2)(m - 1)$ . In the sequel, we will sometime use terms from [2], which deal with disks and sectors and also with more suitable terms to this correspondence, columns and symbols. The original sequence of columns is also modified by placing one of the parity columns in the beginning of the block as will be explained in Section III.

This code corrects all errors if they span one column only in a block. However, when used as a binary nonphased burst-error-correcting code, the burst may spread over two consecutive columns, while maintaining a length of up to one column. In this correspondence, we will show that this code is useful also for handling errors of this type, show a new decoding algorithm, and analyze the probability of success in correcting nonphased bursts.

Another advantage of the algorithm is its simplicity. Both the encoder and the decoder mainly need word XOR operations, which can be performed within one cycle of most microprocessors, and cyclic shift operations that are also one cycle operations per word size. The most efficient choice for  $m$  is one more than the CPU word size. Using this value for  $m$ , one cycle ROR (in assembly—ROtate Right through carry) accomplishes the cyclic shift. The number of calculations in the encoder or in the decoder is in order of  $m^2$ . The number of calculations for a Reed–Solomon code of comparable size is in the order of  $m^3$ . Furthermore, the Reed–Solomon algorithm is much more complex and requires far more processing power when implemented in software. The complexity of the new algorithm is comparable to or even lower than that of cyclic codes using the error trapping decoder, depending

on the order of the generator polynomial. There is about  $m^2$  polynomial divisions needed ( $m^2$  is the code length for the same block size), where the complexity for each polynomial division is proportional to its order.

## II. THE EVENODD SCHEME

### A. The Encoding Scheme

The encoding scheme is taken from the original paper [2] and need not be changed for the new application. Assume there are  $m + 2$  disks, the first  $m$  disks contain the data and the other two are redundant. In the new application, a disk represents a column of bits in the array. If  $m$  is a prime number, the correction of the phased burst is guaranteed. However, even for nonprime  $m$ , the correction failure probability (in short “fail probability”) is not larger than the typical fail probability of the new algorithm, so nonprime values of  $m$  are also considered. To simplify the presentation, we assume that each of the  $m$  disks has  $m - 1$  symbols (sectors) of information on it. The symbol’s size can vary from one bit to any size whatsoever. However, for the new decoding algorithm the symbol is required to be binary.

Let  $\{a_{i,j}\}$ ,  $i = 0, \dots, m - 2$ ,  $j = 0, \dots, m - 1$ , be an array of  $m(m - 1)$  information symbols. Now we define the encoding method. The first step is to append a row of zeros to the end of the array, so now data symbols are indexed 0 to  $m - 2$ , and  $a_{m-1,i} = 0$ . These zero bits are not transmitted, but are defined for convenience. The parity columns indexed  $m$  and  $m + 1$  are defined as follows:

$$s = \bigoplus_{i=0}^{m-1} a_{m-i-1,i} \tag{1}$$

$$a_{j,m} = \bigoplus_{i=0}^{m-1} a_{j,i}, \quad j = 0, \dots, m - 1 \tag{2}$$

$$a_{j,m+1} = s \oplus \left( \bigoplus_{i=0}^{m-1} a_{\langle j-i \rangle_m, i} \right), \quad j = 0, \dots, m - 1 \tag{3}$$

where  $\langle i \rangle_m = i \bmod m$ , and  $\oplus$  denotes XOR. Equations (1)–(3) define the two redundant disks, where disk  $m$  is a simple XOR disk, in which each sector is a XOR of all corresponding sectors in the data disk. Disk  $m + 1$  is calculated using a diagonal XOR of the data disks’ sectors. The temporary variable  $s$  is the parity bit of the diagonal and is not transmitted.

### B. The Original Decoding Scheme

Let  $\{b_{i,j}\}$  be an array of symbols received from the channel where  $\{a_{i,j}\}$  is the original (correct) transmitted array. We define the zero vector  $\bar{0} = (0, 0, 0, \dots, 0)$  and the unity vector  $\bar{1} = (1, 1, 1, \dots, 1)$ . For any vector  $X = (x_0, x_1, \dots, x_{m-1})$  let  $R(X) = (x_{m-1}, x_0, x_1, \dots, x_{m-2})$  be called a left cyclic shift and  $R_j(X)$  is the result of applying  $R(X)$   $j$  times. The decoding operation is performed by adding a row of zeros to the array, and calculating the straight syndrome vector

$$S0_i = \bigoplus_{j=0}^{m-1} b_{i,j}, \quad i = 0, \dots, m - 1 \tag{4}$$

and the diagonal syndrome vector

$$S1_j = b_{j,m+1} \oplus \left( \bigoplus_{i=0}^{m-1} b_{\langle j-i \rangle_m, i} \right), \quad j = 0, \dots, m - 1. \tag{5}$$

Note, that since the internal variable  $s$  of the encoder is unknown, for any error burst, two possibilities for  $S1$  exist, one inverse of the other. Examining  $S0$  and  $S1$  one can determine if the errors are in the data

columns or in the parity columns. In case the data is corrupted, the incorrect column is found by the first  $j$  that satisfies

$$R_j(S_0) \in \{S_1, \bar{1} \oplus S_1\}. \quad (6)$$

Once we have determined the value for  $j$ , the index of the incorrect column, the correction is done using simple XOR operation with  $S_0$  that contain the incorrect symbols. If there is no such  $j$  that satisfies (6), the algorithm declares an uncorrectable error.

### III. THE NOVEL SCHEME

In this correspondence, we suggest a change in the scheme that modifies it for use in a communication channel with burst noise. Let  $P_0$  denote the  $m$ th column of  $\{a_{i,j}\}$  and  $P_1$  shall denote the  $m+1$ th column of  $\{a_{i,j}\}$ , i.e., the parity columns. We shall rearrange the columns to be transmitted in the order  $P_0, 0, 1, \dots, m-1, P_1$ . The bits in each column are transmitted from  $m-2$  to  $0$ . We are interested in correcting bursts of errors with a length of not more than one column (i.e.,  $m-1$  bits). The burst can start at column  $p$  row  $q-1$ , and end (at most) at column  $p+1$  row  $q$ . In this case, the burst spans column  $p$  rows  $0, \dots, q-1$  and column  $p+1$  rows  $q, \dots, m-2$ . The parity  $P_0$  was placed first so that one burst will not affect both parities, resulting in high failure rates. In fact,  $P_0$  can be placed anywhere in the block as long as it is not adjacent to  $P_1$ , without changing the fail probability. The information that we have to retrieve from the syndromes of the received array is as follows.

- 1) The rows which contain an error. This information is found simply in the syndrome vector  $S_0$ .
- 2) The number of the column in which the errors start, denoted by  $p$ .
- 3) The portion of the burst that is in the first of the two columns, denoted by  $q$ . For example, if the length of the column is  $m=11$ , and the error burst span bits  $0-5$  of column  $p$  and bits  $6-9$  in column  $p+1$ , then  $q=6$ . The phased-error case will be the special case of  $q=0$ , where the whole burst is within the column  $p+1$ .

In the original phased-error decoder, a zero bit is added to all the data columns and a cyclic rotation is performed on the syndrome  $S_0$  until it is equal or complementary to syndrome  $S_1$ . The number of the column in which the errors start is determined by the number of rotations. Similarly, in the new algorithm, we use a cyclic rotation operation in order to locate the column number. However, in order to match  $S_0$  and  $S_1$ , the cyclic rotation must be preceded by a correction in  $S_1$ .

If columns  $p$  and  $p+1$  contain errors, it is easy to show that the  $S_1$  syndrome is the union of the error pattern of column  $p$  shifted left  $p$  times,<sup>1</sup> and the error pattern of column  $p+1$  shifted left  $p+1$  times. The error pattern found in  $S_1$ , which may be inverted (depending on the value of the  $s$  bit) is obtained by two steps. In the first step, we insert a "0" at position  $q$  of  $S_0$ , and left-shift the bits starting from this position. We define this operation as

$$T_q(x_0, x_1, \dots, x_{m-2}, 0) = (x_0, x_1, \dots, x_{q-1}, 0, x_q, \dots, x_{m-2}).$$

In the second step the pattern is rotated  $p$  cyclic left shifts. These two steps are shown in Fig. 1, wherein each line in the figure shows the syndrome column as it is modified through the operations above. Fig. 1(a) shows the  $S_0$  syndrome in which bits of value "1" represent rows in error. The vector after the zero insertion is shown in Fig. 1(b), and after rotation in Fig. 1(c).

The first step of the decoding algorithm is to distinguish between three possible cases.

<sup>1</sup>Before shifting left or right one has to transpose the column vector to be a row, and this row is placed in a shift register msb-left, lsb-right.

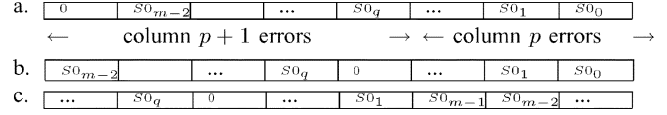


Fig. 1. Generation of  $S_1$  from  $S_0$ .

```

for p = 0 to m - 1
  for q = 0 to m - 2
    S0_NEW = R_p(T_q(S_0))
    if {S0_NEW = S1 or S0_NEW = (\bar{1} \oplus S1)} stop
  end
end

```

Fig. 2. The core of the new decoding algorithm.

- 1) The number of "1's" in  $S_0$  is equal to that of  $S_1$ .
- 2) The sum of the number of "1's" in  $S_0$  and the number of "1's" in  $S_1$  is  $m$ .
- 3) Neither of the above.

In the first case, we proceed to the matching process. The second case indicates that  $S_1$  is inverted ( $s=1$ ) and, therefore, needs to be inverted before the matching process. In the third case, we conclude that the parity columns themselves were corrupted which requires special treatment as follows. We test for the case of a corrupted  $S_1$  by rotating it once to the left, and checking if the bits of  $S_0$  and  $(S_1$  or  $\bar{1} \oplus S_1)$  match. This match is performed from the first "1" in  $S_0$  (going from bit  $m-1$  to bit  $0$ ). We test for the case of a corrupted  $S_0$  by similarly checking if the bits of  $S_0$  and  $(S_1$  or  $\bar{1} \oplus S_1)$  match, starting from the first bit that  $S_1$  is "1" going in the opposite direction (from bit  $0$  to bit  $m-2$ ), where the value of bit  $m-1$  of  $S_1$  indicates whether  $S_1$  needs to be inverted prior to the comparison. The algorithm may be summarized as follows.

- 1) Compute the syndromes and check if there is no error by checking if  $S_0 = \bar{0}$  or  $S_1 = \bar{0}$  or  $S_1 = \bar{1}$ . Otherwise, continue.
- 2) Count the number of "1's" in  $S_0$  and  $S_1$ , and determine whether they are equal or add up to  $m$ . If one of these conditions is satisfied then skip to step 4). Otherwise, either the burst corrupted  $S_0$  and the first data column, or it corrupted the last column and part of  $S_1$ .
- 3) Correct the data according to the corrupted parity case (as explained above) and quit.
- 4) If the sum of the number of "1's" in  $S_0$  and the number of "1's" in  $S_1$  is  $m$  then invert  $S_1$ .
- 5) Scan all the possibilities of  $p$  and  $q$ , apply  $T_q(x)$  and  $p$  left cyclic rotations on  $S_0$  until  $S_1$  is equal or complementary to  $S_0$ .
- 6) Correct the data using  $p$ ,  $q$ , and  $S_0$ .

The algorithm's main task, step 5), is summarized in Fig. 2. Note that various optimizations for complexity reduction can be applied according to the implementation method resulting in a very efficient and compact implementation. For example, the inner loop can be eliminated: In order to compare two words where part of one word is shifted once, we look for the first difference between the two words and then shift the word and compare again, now start from the bit where there was a difference in the first comparison. The number of operations is now in the order of  $m^2$  (i.e.,  $O(m)$  word comparisons).

### IV. AN UPPER BOUND ON THE FAILURE RATE

The decoding failure is when the decoder finds an incorrect  $(p, q)$  pair, which satisfies the matching conditions. Each such failure leads to erroneous correction and therefore a corrupted block. The failure

probability depends on the bursts statistics. We assume that burst position is randomly chosen within the array, burst length is  $m - 1$ , and a 50% bit-error probability within the burst. In a more realistic situation the burst length is exponentially distributed. We will later modify the analysis and give results for shorter bursts, and then any distribution can be evaluated.

We assume that only data columns are corrupted. We will add the case of errors in the parity columns later. For simplifying the analysis, we will treat the case of burst starting in the last data column as continuing cyclically to the first one. Therefore, all possible values for  $(p, q)$  are allowed. If  $m$  is chosen to be odd, there cannot be confusion between a pattern and its inverse (the inverse occurs when  $s = 1$ ). For even  $m$  this case should also be added to the analysis.

The code is a linear binary block code. The length of  $S0$  is  $m$  bits and its first term is always 0. Since the possibility of  $S0$  being all zeros is irrelevant (in this case, there are no errors), then there are  $2^{m-1} - 1$  relevant vectors that define the vector space of  $S0$ . If  $(p, q)$  is the location of an error burst, the condition  $T_q(R_p(S0)) = (S1 \text{ or } \bar{1} \oplus S1)$  will be called a match. In the following lemma, we will prove that the probability that there is more than one match is not dependent on the burst start location. If we assume a random search order when looking for a match, then the decoder has uniform fail probability in terms of burst location.

*Lemma 1:* Let  $(p_0, q_0)$  be the location of a random error burst. The probability that there is more than one possible  $(p, q)$  pair that satisfy a match is not dependent on  $(p_0, q_0)$ .

*Proof:* Suppose that burst occurred in some specific  $(p_0, q_0)$  pair and pattern  $S0$ . Suppose that there are other  $N - 1$  pairs  $(p_i, q_i)$ ,  $i = 1, \dots, N - 1$  that generate the same  $S1$ , which will be called images. We will show that there exists a burst pattern which will be called  $S0'$  that if placed in the first column will have the same number of images.

Let us apply  $(p_0 + q_0 + 1) \bmod m$  right shift rotations to  $S1$ , and call the result  $S1'$ . This is equivalent to letting the burst be at position  $(m - q_0 - 1, q_0)$  and its images at  $((p_i - p_0 - q_0 - 1) \bmod m, q_i)$ . The operation moved the zero that the operation  $T_q$  inserted into  $S1$  to position  $m - 1$ . We can define a new burst pattern by  $S0' = S1'$ . Every  $S1$  that  $S0$  can generate can be generated also by  $S0'$  since  $S0'$  is only a shifted version of  $S0$  (excluding the leading zero). Therefore, the pair  $S0', S1'$  has the same number of images as the pair  $S0, S1$ . Note that setting  $S0' = S1'$  is equivalent to placing the burst at the first column with  $(p = m - 1, q = 0)$ .  $\square$

Let us call the location  $(p = m - 1, q = 0)$  the natural burst and according to the lemma we can assume without loss of generality that the burst occurred in the natural place. For each  $(p, q)$  pair we like to find the burst patterns  $S0$  that if placed in the natural place or if placed in burst start location  $(p, q)$  will generate the same  $S1$ . We can write a set of binary equations to find the number of possible solutions.

We take the vector  $S0 = (w_0, w_1, \dots, w_q, w_{q+1}, \dots, w_{m-1})$  and get the set of  $m$  equations with  $m$  variables

$$R_p(T_q(S0)) = S0. \quad (7)$$

In addition to this equation we also have  $w_{m-1} = 0$ . The rank of this equation set, denoted by  $\text{RANK}_{p,q}$ , and the rank of the vector space of the solutions to this set are summed up to  $m$ . If the rank is full, then there is only one solution: the zero vector. This means that there is no pattern that can be confused between the natural location and the burst start location  $(p, q)$ . For a different rank, the number of such patterns is

$$2^{m - \text{RANK}_{p,q}} - 1. \quad (8)$$

For example, let  $m = 5$ ,  $q = 1$ , and  $p = 3$ . The original vectors, before applying  $R_p$  and  $T_q$  are

$$S0 = (w_0, w_1, w_2, w_3, w_4) \quad (9)$$

$$S1 = (w_0, w_1, w_2, w_3, w_4). \quad (10)$$

The effect of  $q$  (which determines where the bits start in the second column) is shifting  $w_{m-1}$  to the  $q$ th place and shifting every bit to its left, one space to the left,<sup>2</sup> giving us the next set

$$S0 = (w_0, w_1, w_2, w_3, w_4) \quad (11)$$

$$S1 = (w_0, w_4, w_1, w_2, w_3). \quad (12)$$

The effect of  $p$  is adding  $p$  cyclic left rotations, leaving us with the next set

$$S0 = (w_0, w_1, w_2, w_3, w_4) \quad (13)$$

$$S1 = (w_1, w_2, w_3, w_0, w_4). \quad (14)$$

This gives us the following set of equations:

$$w_0 = w_1; w_1 = w_2; w_2 = w_3; w_3 = w_0; w_4 = w_4. \quad (15)$$

Since the number of all the possibilities of  $S0$  is  $2^{m-1} - 1$ , and there is a probability of  $1/2$  that the true location will be encountered when the decoder search is independent of the burst location, then the error probability for a specific  $(p, q)$  pair is

$$P_{\text{fail}}(p, q) = \frac{1}{2} \frac{2^{m - \text{RANK}_{p,q}} - 1}{2^{m-1} - 1}. \quad (16)$$

Now we repeat the calculation for all pairs of  $(p, q)$  and use the union bound to upper-bound the fail probability

$$P_{\text{fail}} \leq \sum_{p=1}^{m-2} \sum_{q=0}^{m-2} P_{\text{fail}}(p, q). \quad (17)$$

Note that the cases  $p = 0$  and  $p = m - 1$  were excluded since they represent images that are identical to the correct burst. A tighter bound will be obtained later in this section.

The next problem is to find an easy and fast way to calculate the rank of the equation sets. Since we are considering sets of  $m$  equations with  $m$  variables, the complexity of the solution is of order  $m^2$ . We present here a scheme that allows the calculation of the rank with a complexity of order  $m$ . The algorithm is described as follows.

We start with the equation that contains  $w_0$  in the right-hand side. We take the variable from the left side of this equation and find the equation that has this variable in its right side. We take the variable from the left side and find again the equation that has it in the right side. We repeat this procedure until completing a cycle. For our example, the cycles are

$$w_0 \rightarrow w_3 \rightarrow w_2 \rightarrow w_1 \rightarrow w_0 \quad (18)$$

$$w_4 \rightarrow w_4. \quad (19)$$

If we did not traverse all the variables, we start another cycle at the first variable that we did not traverse. The rank is given by

$$\text{RANK}_{p,q} = m - L_{p,q} + 1 \quad (20)$$

where  $L_{p,q}$  is the number of cycles. The explanation is simple. Each cycle group variables that are equal and present one free variable in

<sup>2</sup>The direction left is defined as the direction of increasing indices as common in binary representation of a number.

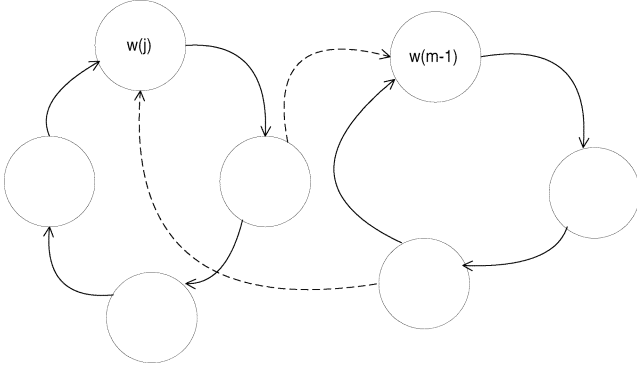


Fig. 3. Joining two loops.

the solution. Therefore, the number of cycles is the rank of the solution space. However, the cycle that contain the  $w_{m-1}$  is not free since  $w_{m-1} = 0$ , and so adding this limitation the rank of the solution space is (number of cycles)  $- 1$ . The rank of the equations set plus the rank of the solution space is the number of variables  $m$ .

Using the following lemma, we will now obtain a (much) tighter upper bound by deleting solutions that are completely included in other equation sets.

**Lemma 2:** If  $L_{p,q} > L_{p,q\pm 1}$  then any  $S_0$  solving the equations for  $(p, q \pm 1)$  will solve also the set for  $(p, q)$ .

*Proof of Lemma 2:* One can verify that increasing or decreasing  $q$  by 1 switches the location of  $w_{m-1}$  with the adjacent variable. Now, if that variable belongs to a different loop than that containing  $w_{m-1}$ , switching the places links these two loops together to one loop, as shown in Fig. 3, thus, the rank is increased. Otherwise, the switching breaks the loop that contains  $w_{m-1}$  and that variable into two loops, thus, the rank is decreased. In the first case, uniting the two loops causes the loop containing  $w_{m-1}$  to increase, which means more variables are forced to be 0. As a result, any solution for the system with the united loop is also solution of the case where the loop is broken into two loops.  $\square$

Using Lemma 2, the local maxima of the rank in a column cover all solutions contained in its neighbors and these can be excluded from the union bound. The updated bound is

$$P_{\text{fail}} \leq \sum_{p=1}^{m-2} \sum_{q=0, L_{p,q} > L_{p,q+1} \& L_{p,q} > L_{p,q-1}}^{m-2} P_{\text{fail}}(p, q) \quad (21)$$

where the condition  $L_{p,q} > L_{p,q-1}$  is taken as true if  $q = 0$ .

#### A. Calculating a Lower Bound for the Scheme's Performance

The bound we present in this subsection is a lower bound to the algorithm performance. The results show that the bound becomes tighter as  $m$  increases. Going through the cases in which the scheme cannot correct the error burst shows that in the majority of these cases there are special patterns for the  $S_0$  vector, cases in which  $S_0$  is divided to recurring patterns as shown in the following example.

Let us assume  $m = 11$  and

$$S_0 = (S_{0_0}, \dots, S_{0_{m-1}}) = (1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0).$$

We can see that this is the pattern  $(1, 0, 1, 1, 1)$  is repeating itself, followed by the additional 0. If we choose  $q = 5$ , we move the zero after the first repetition and we get  $(1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1)$ . Rotation

to bring the zero to position 10 will cause the patterns to switch positions, but since they are equal, we get the same original vector.

The fail probability for a periodic pattern with period  $(m-1)/N$  is  $(N-1)/N$  since there are at least  $N-1$  images to that pattern, and the decoder search in random order. We take another example, in which  $m = 5$ . There are 15 possibilities for  $S_0$ . Out of these, there are three cyclic patterns:

- 1)  $S_0 = (1, 0, 1, 0, 0)$  or  $(0, 1, 0, 1, 0)$ , with  $N = 2$ ;
- 2)  $S_0 = (1, 1, 1, 1, 0)$ , with  $N = 4$ .

Now we calculate the total probability

$$P_{\text{fail}} \geq \frac{3/4 + 1/2 + 1/2}{15} = 0.116. \quad (22)$$

There are also patterns that are not cyclic and have images, so the equation is a lower bound. However, the results show that the ratio between the number of such patterns to the cyclic ones is diminishing for large  $m$ , so the lower bound tightens as  $m$  increases.

Since the  $m$ th symbol of  $S_0$  is always 0, and that is the symbol that is moved to the  $q$ th location in the vector, we can find two kinds of patterns:

- 1) patterns that are periodic for the  $m-1$  bits of  $S_0$ , requiring  $q > 1$  for creating images;
- 2) patterns that are periodic for all the  $m$  bits of  $S_0$  (including the preceding zero), and then the symmetry is obtained using only cyclic rotations.

For a prime  $m$  the second case is not possible. We begin with finding the symmetry for the first case, the second case is very similar. First, we find all the prime dividers of  $m-1$ . All the nonprime divider cases are included in the prime dividers case, though with higher  $(N-1)/N$  factor. Using the prime dividers only is therefore a lower bound (for large  $N$  the difference is minor). Looking at  $m = 7$ , for instance, the possible dividers are 2 and 3 and we can find symmetrical patterns that include three vectors of 2 bits or two vectors of 3 bits. We denote the dividers by  $\{N_j\}$ . The number of all the possible patterns for a given  $N_j$  is  $2^{\frac{m-1}{N_j}} - 1$ . For a nonprime  $m$ , we should also include the patterns that are periodic in  $m$ . Let  $\{N'_j\}$  be the prime dividers of  $m$  if one exists. The bound formula is therefore,

$$P_{\text{fail}} \geq \sum_j \frac{N_j - 1}{2^{m-1} - 1} \left( 2^{\frac{m-1}{N_j}} - 1 \right) + \sum_j \frac{N'_j - 1}{2^{m-1} - 1} \left( 2^{\frac{m}{N'_j}} - 1 \right). \quad (23)$$

The value of  $N_j = 2$  will always exist in one of the summations, and lead to the highest term. We can bound the expression by

$$P_{\text{fail}} \geq 2^{\frac{-1-m}{2}}. \quad (24)$$

For large  $m$ , this bound is tightening and becomes a good approximation since the ratio between the term resulting from  $N_j = 2$  to all the other terms is increasing. The results for all the bounds compared to simulation are shown in Fig. 4. Note that the results in which errors in the parity columns are included are lower since the fail probability when the parities are affected is lower than if the data only is affected as will be discussed shortly. Since the errors are placed uniformly, fewer bursts  $(m/(m+2))$  are affecting the data, leading to an overall (slight) decrease in the fail probability.

#### B. Special Cases and Corrections

There are few cases for the error burst that are not covered by the preceding derivations and require small modifications in the formula.

1) *An Error Between Two Blocks:* If the information blocks (that include  $m$  columns each) are transmitted successively, then the incorrect column could start at the last column of a specific block and end

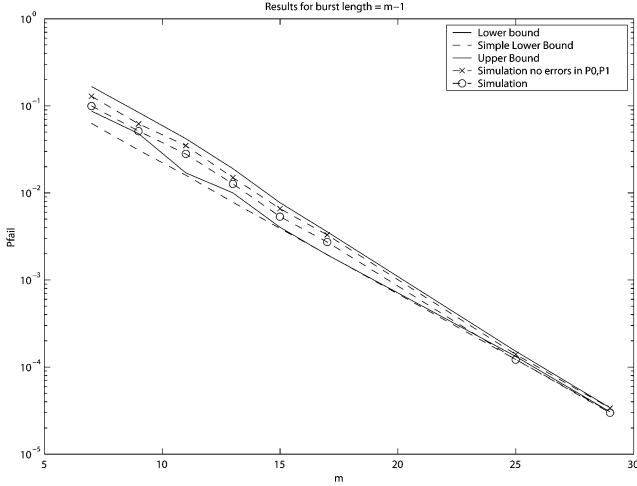


Fig. 4. Simulation with and without errors in the parity, lower bound (23) and simple lower bound (24) and upper bound (21).

at the first column of the next block. In this situation, the first block contains errors only in the last column, which is the  $P0$  vector. Since  $S1$  is zero (or all ones) we know that the data is correct. For the second block, all the errors are in  $P1$  and again  $S0$  is  $\bar{0}$  and the data is correct.

2) *Errors in the Syndrome Vectors:* We treat four different cases.

- 1) All the errors are in  $P0$ .
- 2) All the errors are in  $P1$ .
- 3) The errors are divided between the last data column (column  $m - 1$ ) and  $P1$ .
- 4) The errors are divided between the first data column (column 0) and  $P0$ .

For the first two cases, since one of the syndromes is zero (or all ones) there is no problem, except for the case where the error burst is all ones which occur in probability  $2^{(1-m)}/m/(m+2)$  which is very small compared to other fail probabilities and can be ignored.

For cases 3) and 4), the solution is described in step 3) of the decoding algorithm and its success is almost guaranteed, except for the cases where the number of "1's" match even though  $S0$  or  $S1$  are corrupted, which can be possible due to the possible inversion of  $S1$ , and for cases there is a confusion between case 3) and case 4). Between the two failure causes the first is the more significant. These failure cases are not very difficult to analyze, but since their contribution to the overall error probability is much lower than the regular bursts in data, we did not calculate analytically bounds for these cases.

3) *The Scheme Performance for Short Bursts:* As the burst length decreases the failure rates drop rapidly. In the burst model we use, there is a random starting point within the block and the burst length is  $l$  bits which are in error with probability  $1/2$ . At burst length of  $l = (m - 1)/2$  we get 100% error detection and correction. In this case, since the burst is shorter than half a column, one can verify that there is no two cyclic rotations that will give the same  $S1$  syndrome. Cases 3) and 4) of Section IV-B2 are not possible, since it is not possible to have the number of "1's" in  $S0$  and  $S1$  neither equal nor summed to be equal to  $m$ . The explanation is as follows (proof for corrupted  $S1$ , the other case is similar). Let  $j$  be the number of errors in the corrupted data column,  $k$  be the number of errors in the corrupted parity column  $P1$ . The resulting number of "1's" in  $S0$  is  $j$  and in  $S1$  is  $j + k$  or  $m - j - k$ . Neither  $j = j + k$  nor  $m - j - k = j$  is possible under the condition  $j + k \leq l \leq (m - 1)/2$  for  $j, k > 0$ .

The bound for the case  $(m - 1)/2 < l \leq m - 1$  is similar to the union bound for the full burst length and is given by

$$P_{\text{fail}} \leq \sum_{p=1}^{m-2} \sum_{q=0, L_{p,q} > L_{p,q+1} \& L_{p,q} > L_{p,q-1}} P_{\text{fail}}(p, q, l) \quad (25)$$

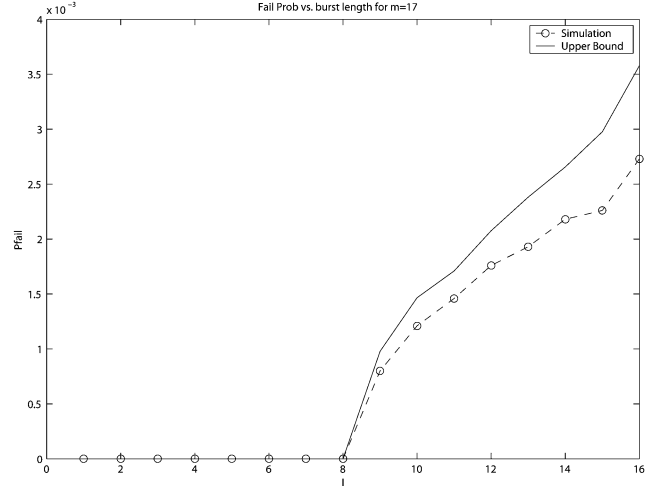


Fig. 5. Simulation with errors in the parity and upper bound (25), results shown for  $m = 17$ .

where  $P_{\text{fail}}(p, q, l)$  is computed according to (16), but the computation of the rank is modified as follows. Since  $m - l$  bits in  $S0$  must be zero, and we can place the burst anywhere we want, we place it such that the last  $m - l$  bits are 0. Therefore, we set the variables  $w_1, \dots, w_{m-1}$  to zero in the equation set. We add another counter to the algorithm of loop counting which counts the loops that contain at least one of these variables. Let us denote this count by  $M_{p,q}$ . Now the rank can be computed by

$$\text{RANK}_{p,q} = m - L_{p,q} + M_{p,q} \quad (26)$$

Results for the bound and the simulation for  $m = 17$  as an example are shown in Fig. 5. We can see that the bound is very tight, and that the fail probability moderately decreases with burst length for a given  $m$  as long as it is above  $(m - 1)/2$ .

## V. CONCLUSION

We presented a novel scheme for handling nonphased burst errors. This scheme is based on an algorithm from the RAID field in the computer world, which handles phased burst errors only. Our scheme does not satisfy the formal definitions of burst-error-correcting code, but yet, we showed that the rate of failure, the probability of having an uncorrected burst, is decreasing exponentially with the square of the size of the block. We developed a simple decoding algorithm to correct nonphased burst errors for this code. The code has low redundancy ( $l = (n - k)/2$ ), it is very simple, it is systematic, and it needs only  $O(m^2)$  operations to decode in a very easy to implement software or hardware. The scheme performance was given both with tight bounds and approximately in a simple closed form. This code provides a practical alternative to the well-known burst-error-correcting codes.

## REFERENCES

- [1] J. G. Proakis, *Digital Communications*, 3rd ed. New York: McGraw-Hill, 1995.
- [2] 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, no. 2, pp. 192–201, Feb. 1995.
- [3] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. New York, NY: North-Holland, 1977.
- [4] M. Blaum, "A class of byte-correcting array codes," IBM Res. Rep., RJ 5652 (57151), 1987.
- [5] M. Blaum and R. Roth, "New array codes for multiple phased error correction," *IEEE Trans. Inf. Theory*, vol. 39, no. 1, pp. 66–77, Jan 1993.
- [6] R. Goodman and M. Sayano, "Size limits on phased burst error correcting array codes," *Electron. Lett.*, vol. 26, pp. 55–56, 1990.

- [7] R. Goodman, R. J. McEliece, and M. Sayano, "Phased burst correcting array codes," *IEEE Trans. Inf. Theory*, vol. 39, no. 2, pp. 684–693, Mar. 1993.
- [8] S. Lin and D. J. Costello Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.
- [9] W. Zhang and J. K. Wolf, "A class of binary burst error-correcting quasi-cyclic codes," *IEEE Trans. Inf. Theory*, vol. 34, no. 3, pp. 463–479, May 1988.
- [10] O. Keren and S. Litsyn, "Codes correcting phased burst errors," *IEEE Trans. Inf. Theory*, vol. 44, no. 1, pp. 416–420, Jan. 1998.
- [11] R. Roth and G. Seroussi, "Reduced-redundancy product codes for burst error correction," *IEEE Trans. Inf. Theory*, vol. 44, no. 4, pp. 1395–1406, Jul. 1998.
- [12] M. Blaum, "A family of efficient burst-correcting array codes," *IEEE Trans. Inf. Theory*, vol. 36, no. 3, pp. 671–675, May 1990.
- [13] R. G. Gallager, *Information Theory and Reliable Communication*. New York: Wiley, 1968.

## Improved List Decoding of Generalized Reed–Solomon and Alternant Codes Over Galois Rings

Marc A. Armand, *Member, IEEE*

**Abstract**—We present a two-stage list decoder comprising an errors-only Guruswami–Sudan (GS) decoder and an errors-and-erasures GS decoder as component decoders in the first and second stage, respectively. The two stages are coupled via a post-processor which selects a codeword from the output list of the first component decoder, from which erasure locations are obtained for the second stage. When applied to a generalized Reed–Solomon (RS) code over a Galois ring  $R$  that maps into a generalized RS code of the same length  $n$  and minimum (Hamming) distance  $d$  over the corresponding residue field, the proposed decoder exploits the presence of zero divisors in  $R$  to correct  $s$  errors where

$$w = \left\lceil n - \sqrt{n(n-d)} - 1 \right\rceil$$

$$< s \leq \left\lceil n - \sqrt{(n-w)(n-d)} - 1 \right\rceil$$

with a probability determined by  $s$ ,  $w$ , and the ratio of the number of nontrivial zero divisors to the number of units in the code alphabet. Focusing primarily on alternant codes over  $\mathbb{Z}_{2^l}$ , an important class of subring subcodes of generalized RS codes over  $\text{GR}(2^l, \alpha)$ , we demonstrate that the GS decoding radius  $w$  can be exceeded by a substantial margin with significant probability.

**Index Terms**—Alternant codes, Galois rings, generalized Reed–Solomon (RS) codes, list decoding, zero divisors.

### I. INTRODUCTION

In [1], [2], we showed that generalized Reed–Solomon (RS) codes over any commutative ring with identity, may be list decoded using the Guruswami–Sudan (GS) decoder of [3]. For a generalized RS code of length  $n$  and minimum distance  $d$ , that decoder can correct up to

$$w = \left\lceil n - \sqrt{n(n-d)} - 1 \right\rceil$$

Manuscript received July 7, 2004; revised November 9, 2004. The material in this correspondence was presented in part at the IEEE International Symposium on Information Theory, Chicago, IL, June/July 2004.

The author is with the Communications and Information Engineering Group, Department of Electrical and Computer Engineering, National University of Singapore, Singapore, 117576 (e-mail: eleama@nus.edu.sg).

Communicated by M. P. Fossorier, Associate Editor for Coding Techniques. Digital Object Identifier 10.1109/TIT.2004.840901

errors. We refer to this bound as the GS radius. Although this represents a significant improvement over the classical  $\left\lfloor \frac{d-1}{2} \right\rfloor$  bound, it is natural to ask if generalized RS codes over rings can offer any improvement over their field counterparts in terms of the maximum number of errors that can be corrected. If we make a distinction between error values which are units in the code alphabet, and those which are zero divisors, then indeed they can. In this paper, we propose a two-stage list decoder to be applied to generalized RS codes over a Galois ring  $R = \text{GR}(q = p^l, a)$  where  $l \geq 2$ ,  $a \geq 1$ , and comprises of two component GS decoders to make that distinction. The first component decoder coupled to a post-processor, operates over the residue field  $F = \text{GF}(p^a)$  of  $R$  to determine the location of errors in a received word  $y$  which are units in  $R$  while the second, interprets this information as erasure locations and determines the remaining errors in  $y$  which are zero divisors of  $R$  as well as the "erased" values. What makes this decoder work is the fact that a generalized RS code over  $R$  maps into a generalized RS code of the same length and minimum distance over  $F$ .

Our proposed decoder is capable of exceeding the GS radius and corrects  $s$  errors where

$$w < s \leq \left\lceil n - \sqrt{(n-w)(n-d)} - 1 \right\rceil$$

with a probability determined primarily by the ratio  $\gamma$  of the number of nontrivial zero divisors to the number of units in the code alphabet. Thus, for an ensemble of equal-length generalized RS codes over  $R$ , the margin by which our two-stage decoder can exceed the GS radius increases as code rate decreases. Nevertheless, it turns out that our decoding scheme is more effective when applied to alternant codes over  $\mathbb{Z}_q$  (an important class of subring subcodes of generalized RS codes over  $R$ ), than when applied to their parent codes. By construction, to obtain longer generalized RS codes for a fixed prime  $p$ ,  $\gamma$  needs to be reduced. Consequently, the probability of correcting beyond  $w$  errors decreases as  $n$  increases such that for practical values of  $n$ ,  $\gamma$  is too small for the presence of zero divisors in the code alphabet to be exploited. On the other hand, one can increase the length of an alternant code over a fixed alphabet without affecting  $\gamma$ . As such, alternant codes do not suffer the above shortcoming of their parent codes and consequently, significant improvements in their error-correcting capability may be realized, provided  $\gamma$  is sufficiently large, e.g.,  $\gamma = 0.5$ . To illustrate, when applied to three extended narrow-sense Bose–Chaudhuri–Hocquenghem (BCH) codes over  $\mathbb{Z}_4$  of length 1024 and rate  $1/4$ ,  $1/2$ , and  $3/4$ , a single-stage errors-only GS decoder can correct all error patterns of (Hamming) weight not exceeding 113, 59, and 26, respectively. In contrast, our two-stage decoder can correct up to 150, 75, and 31 errors, respectively, with probability at least 0.99, assuming that i) the probability of error patterns occurring depend only on their (Hamming) weight, and ii) the post-processor always picks the correct codeword from the first component decoder's output list, whenever the number of errors which are units does not exceed the corresponding GS radius.

We proceed with a brief review of generalized RS codes over Galois rings and their subring subcodes, followed by the GS decoder. We shall be using some well-known facts about Galois rings. For more details and proofs, we refer the reader to [8].

### II. PRELIMINARIES

#### A. Generalized RS Codes and Their Subring Subcodes

Let  $R = \text{GR}(q = p^l, a)$  where  $p$  is prime,  $l, a \geq 1$ , and  $F$  its residue field  $\text{GF}(p^a)$  as before. By  $M$ , we denote the maximal ideal in  $R$  which contains all the zero divisors of  $R$ . Recall that  $M$  is generated by  $p$ . Suppose  $\beta$  generates the cyclic group of units of order  $n^* = p^a - 1$  in  $R$ . Then  $\bar{\beta}$  is primitive in  $F \setminus \{0\}$  where  $\bar{f}$  denotes the