Some Specific Features in the Construction of p-ary Reed-Solomon Codes for an Arbitrary Prime p

Zhaneta Savova
Department of Computer Systems and Technologies
National Military University
Shumen, Bulgaria
zh.savova@yahoo.com

Rosen Bogdanov
Department of Communication Networks and Systems
National Military University
Shumen, Bulgaria
r61@abv.bg

Abstract. **The Reed-Solomon (RS) codes, proposed in 1960 by Irving Reed and Gustav Solomon as a subset of error-correcting codes, have many current applications. The most significant of which are data recovery in storage systems, including hard drives, minidiscs, CDs, DVDs, Google's GFS, BigTable, and RAID 6, as well as in communication systems such as DSL, WiMAX, DVB, ATSC, and satellite communications. Additionally, RS codes are used as Bar codes in management and advertising systems, such as PDF-417, MaxiCode, Datamatrix, QR Code, and Aztec Code. Nowadays, RS codes over Galois Fields GF(2^m) with base 2 are commonly used in these applications, with the GF(2^8) field being the most widely used. This allows all 256 values of a byte to be represented as a polynomial with 8 binary coefficients over GF(2^8). Considering RS codes as cyclic codes in GF(2^m) fields, as well as the validity of mathematical dependencies in arbitrary field GF(p^m), is a motivation to verify and generalize the idea of generating RS codes in a field with base other prime than 2. As a result, the paper derives the specific features of the construction of Reed-Solomon codes by considering them as a family of codes over any field GF(p^m) whose base is a prime *p* other than 2. The paper also discusses the unique properties of basic arithmetic operations in the arbitrary field GF(p^m), which arise from the non-uniqueness of the inverse elements *a* and *-a* in a field with base other than 2.**

Keywords: BCH Codes, Error Correcting Codes, Extended Galois Field, p-ary Reed-Solomon Codes.

I. INTRODUCTION

The Reed-Solomon (RS) codes are non-binary cyclic error correction codes proposed by Irving Reed and Gustav Solomon in 1960 [1]. By adding symbols to check the data, the RS code can detect any combination of up to a maximum of t erroneous symbols or correct up to $\lfloor t/2 \rfloor$ symbols. Here $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. Using the RS code as an erasure correction code, it can correct up to a maximum of t known erasures, or it can detect and correct combinations of errors and erasures. An advantage of RS codes is that they are suitable for correcting burst errors because a sequence of $m + 1$ bit errors can affect at most two symbols of size m.

Reed-Solomon codes are most commonly used in data storage systems to correct packet errors caused by media defects. The information recorded on a compact disc (CD) is divided into segments called frames, with each frame containing 24 information symbols. Each symbol is represented as an element of the Galois field GF(2^8). The code used to correct errors is called a Cross-Interleaved Reed-Solomon code (CIRC) because it is obtained by a cross-interleaving process of two shortened Reed-Solomon codes [2], [3]. The first code C1, which uses symbols from GF(2^8) as input information, is the shortened Reed-Solomon code (32, 28). The second code C2 is a shortened (28, 24) Reed-Solomon code again operating in the GF(2^8). This sets the rate of the CIRC code to r = 24/32 = 3/4 = 0,75. Both codes C1 and C2 have a minimum distance $d_{min} = 5$, which determines their ability to correct up to a maximum of 2 errors per codeword or to perform up to a maximum of 5 erasure corrections.

Digital Video Discs (DVDs) use a similar error correction scheme called Reed-Solomon Product Code (RS-PC) [4]. In it, the two truncated Reed-Solomon codes C1 and C2 are relatively longer than those in CIRC, being (208, 192) and (182, 172) respectively. In addition, the RS-PC rate is much higher $r = 172.192/(182.208) = 0.872$. Blu-ray Discs use an error correction system with an efficient way of indicating packet errors called Picket Code (PC). Pickets are special

Print ISSN 1691-5402
Online ISSN 2256-070X
https://doi.org/10.17770/etr2024vol4.8212

columns inserted at regular intervals between the columns of main data. The underlying data is protected by an efficient Reed-Solomon code. The pickets are protected by a second, independent and extremely powerful Reed-Solomon code. During decoding, the pickets are first corrected, and the information obtained can be used to calculate the location of possible errors in decoding the underlying data.

As a result of the use of error correction codes in data storage systems, the maximum packet error length is about 500 bytes for the CIRC code, 2200 bytes for the RS-PC code, while for the PC it is about 9900 bytes. The DVD RS-PC code can reduce the random input error from 2.10^{-2} to a data error of 10^{-15}, which is about 10 times better than the CD [4]. Under the same conditions, the possible error in the data on an optical disc using the Pickett code is $1{,}5.10^{-18}$, while on an optical disc using RS-PC it is $5{,}7.10^{-7}$ [5].

Most two-dimensional barcodes, such as QR Code, PDF-417, MaxiCode, Aztec Code, etc., use Reed-Solomon codes to correct errors if part of the barcode is damaged [6]. Modern data transmission systems used in digital television, satellite space and wireless communications use specialized concatenated codes, one of which is the Reed-Solomon code [7], [8].

Erasure-coded storage using Reed-Solomon codes is now widely used in large, distributed storage systems, including Google File System (GFS), Facebook Hadoop Distributed File System (HDFS), Windows Azure storage, and data centers [9], [10], [11].

Nowadays, multi-level signals and sequences are emerging as a prominent feature in today's high-speed communication systems. New methods such as the PWAM signaling scheme [12], 4D PAM-7 [13], and Automotive Ethernet [14] are used in in-vehicle networking. These methods utilize seven-level pulse amplitude modulation (PAM-7), Four-Dimensional Five Level Pulse Amplitude Modulation (4D-PAM-5), and PAM-3 symbols to improve data transfer rates compared to wire infrastructures.

Real-world applications of Reed-Solomon codes mostly use a Galois field representation of $GF(2^8)$ symbols [15]. In many research papers, the theoretical construction of Reed-Solomon codes is presented over an arbitrary field $GF(q)$ of q elements, where q is a power of a prime number. Despite this theoretical representation, the examples explaining Reed-Solomon codes are over fields of base 2. Therefore, to ensure the error correction features of new multi-level sequences, advanced methods are required to generate not only binary but also nonbinary symbols. The aim of this article is to provide a detailed discussion on constructing Reed-Solomon codes as a family of codes over an arbitrary Galois field $GF(p^n)$. It also highlights the unique characteristics of the codes when operating over finite fields of base other prime than 2.

II. MATERIALS AND METHODS

There are two methods for the construction of Reed-Solomon codes. The first method views the codeword as a sequence of values proposed in Reed and Solomon's original 1960 paper “Polynomial codes over certain finite fields” [1]. The second method views Reed-Solomon codes as Bose-Chaudhuri-Hocquenghem (BCH) codes [16], where the codeword is represented as a sequence of coefficients.

A. Original Presentation of Reed-Solomon Codes

Reed and Solomon consider a field K of degree n over a field Z_2 of 2 elements [1]. They propose a code E by which each k-tuple $(a_0, a_1, \ldots, a_{k-1})$ of K is matched by a 2^n-tuple $(m(0), m(\alpha), m(\alpha^2), \ldots, m(1))$ of K. Here $m(x)$ is a polynomial of degree $k - 1$

$$m(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{k-1}x^{k-1}, \qquad (1)$$

where $a_i, \in K$, $k < 2^n$, and α is the primitive n–th unit root in K. Here, the k-tuple represents the encoded message and the and the 2^n-tuple represents the transmitted message. The authors prove that this code corrects $(2^n - k)/2$ or $(2^n - k - 1)/2$ symbols, depending on whether k is an even or odd number.

B. BCH Representation of Reed-Solomon Codes

Instead of sending al`l values of message polynomial $m(x)$ (1), the transmitter computes another polynomial $s(x)$ of degree at most $n - 1$ (where $n = q - 1$) and sends the n coefficients of this polynomial. The polynomial $s(x)$ is obtained by multiplying the message polynomial $m(x)$ of degree at most $k - 1$ by the generator polynomial $g(x)$ of degree $n - k$, which is used in the transmitter and receiver of the coding system.

The generating polynomial $g(x)$ is defined as a polynomial whose roots are the elements $\alpha, \alpha^2, \ldots, \alpha^{n-k}$ of the field K. Thus, $g(x)$ can be expressed as:

$$\begin{aligned} g(x) = (x - \alpha)(x - \alpha^2) \ldots (x - \alpha^{n-k}) = \\ = g_0 + g_1x + \cdots \\ + g_{n-k-1}x^{n-k-1} + x^{n-k}. \end{aligned} \qquad (2)$$

The transmitter sends $n = q - 1$ polynomial coefficients

$$s(x) = m(x)g(x). \qquad (3)$$

The receiver considers the received symbols as coefficients of the polynomial $r(x)$. If there are no transmission errors ($r(x) = s(x)$), then by dividing the polynomial $r(x)$ by $g(x)$ the message polynomial $m(x)$ can be obtained

$$\frac{r(x)}{g(x)} = m(x). \qquad (4)$$

If transmission errors occur, the division will produce a remainder $e(x)$ with a lower degree than that of $g(x)$, indicating the presence of errors, i.e.

$$r(x) = m(x).g(x) + e(x). \qquad (5)$$

If there is an error $e(x) \neq 0$, the receiver can calculate $r(x)$ for all roots of $g(x)$. This will result in a system of equations that will determine which coefficients have errors and the values of those errors. The Berlekamp-Messy algorithm [17] or extended Euclidean algorithm [18], and the parity polynomial

$$h(x) = (x - \alpha^0)(x - \alpha^{n-k+1}) \ldots (x - \alpha^{n-1}) \qquad (6)$$

$$= h_0 + h_1 x + \cdots + h_{k-1} x^{k-1} + x^k$$

are used to accomplish this.

C. Key Features of Reed-Solomon Codes

Reed-Solomon codes are linear block cyclic (n, k) codes of length n and size k. The minimum Hamming distance of the RS (n, k) code is

$$d_{min} = n - k + 1. \quad (7)$$

For an arbitrary systematic (n, k) code is satisfied

$$d_{min} \leq n - k + 1, \quad (8)$$

since in any (n, k) code, a non-zero codeword with a weight of at most $n - k + 1$ can always be created by resetting all but one of the k information digits.

The code executed with equality at (8) is called Maximum Distance Separable (MDS). As shown in (7), all RS codes are MDS. MDS codes have special properties because they have the maximum possible d_{min} at their length n and size k. For instance, any set of k columns of their generator matrix **G** are linearly independent, meaning any k positions in the block can be used as information. Additionally, the weight distribution of MDS codes can be easily determined.

The RS code's correction capabilities are determined by its minimum distance, d_{min}. It can correct up to $t = (n - k)/2$ erroneous symbols. If the error locations are known in advance, as defined by the erasure term, the RS code can correct up to a maximum of $2t$ erasures. The RS code is capable of correcting any combination of errors and erasures, provided that they fall within its correction capabilities

$$2e + s \leq n - k, \quad (9)$$

where e is the number of errors and s is the number of erasures in the block.

RS codes cannot be binary since the length n of the RS code is smaller than the size of the encoding alphabet. RS codes over $GF(2^m)$ are of particular interest. For instance, for $m = 8$, RS codes can have a length of $n = 2^8 - 1 = 255$ symbols, each 8 bits long.

Since GF(2^m) is a vector space of size m over GF(2), each element of GF(2^m) can be represented by m bits, which are coefficients in the linear combination of selected basis vectors. The element 0 is represented by a zero binary m-tuple (0, 0, ..., 0), regardless of the chosen basis elements. When using this representation, the (n, k) code over GF(2^m) becomes a binary (mn, mk) code with a minimum distance d'_{min} at least as large as the minimum distance of the code it forms. This is because each nonzero element of GF(2^m) has at least one '1' in its binary m-tuple representation. The case where $d'_{min} > d_{min}$ is also possible. Therefore, equivalent binary codes are useful for their burst error correction property. Each burst of errors $(t - 1)\,m + 1$ or fewer consecutive bits will appear as at most t errors in the $GF(2^m)$ symbols. Thus, a decoding algorithm for $GF(2^m)$ code that corrects all combinations of t or fewer errors will also automatically correct all bursts of consecutive errors of length less than or equal to $(t - 1)\,m + 1$ bits. Reed-Solomon codes are ideal for correcting burst errors due to their largest possible d_{min}, given their length $n = 2^m - 1$ (or a divisor of it) and size k ($1 \leq k < n$).

Next, we present an algorithm for generating a family of RS codes over an arbitrary field $GF(p^m)$, where p is an arbitrary prime number. For brevity, these codes will be referred to as p-ary Reed-Solomon codes or pRS codes.

III. Results and Discussion

This section provides the mathematical background of an extended Galois field GF(p^n) and the main steps of the proposed algorithm. It also specifies the algorithm's features for prime p greater than 2 and give some results from algorithm's testing.

A. Mathematical Background of an Extended Galois Field $GF(p^n)$

The field GF(q) is an extension of the Galois Field GF(p) with a power of n if the order of GF(q) can be expressed as a power of the prime p ($q = p^n$), where n is a positive ($n \geq 2$). In these cases, we use the notation Extended Galois Field GF(p^n).

To create an Extended Galois Field GF(p^n), select an irreducible polynomial $p(x)$ over GF(p) [19]. Let α be a root of p(x) such that $p(\alpha) = 0$. The elements of the field GF(p^n) are polynomials of degree $n - 1$, which belong to the ring GF(p)[x] and have coefficients in GF(p)

$$GF(p^n) = \{a_{n-1}\alpha^{n-1} + \cdots + a_1\alpha + a_0 \mid a_i \in GF(p)\}. \quad (10)$$

Arithmetic in GF(p^n) involves polynomial arithmetic modulo the irreducible polynomial $p(x)$. The two main algebraic operations are addition (13) and multiplication (14), which are defined for two elements $f(\alpha)$ (11) and $g(\alpha)$ (12) of GF(p^n).

$$f(\alpha) = \sum_{i=0}^{n-1} a_i \alpha^i = \quad (11)$$
$$= a_{n-1}\alpha^{n-1} + \cdots + a_1\alpha + a_0$$

$$\mathrm{g}(\alpha) = \sum_{i=0}^{n-1} b_i \alpha^i = \quad (12)$$
$$= b_{n-1}\alpha^{n-1} + \cdots + b_1\alpha + b_0$$

Addition in GF(p^n):

$$\big(f(\alpha) + \mathrm{g}(\alpha)\big) = \quad (13)$$
$$= \sum_{i=0}^{n-1} [(a_i + b_i) \bmod p].\, \alpha^i \in GF(p^n).$$

Multiplication in GF(p^n):

$$r(\alpha) = f(\alpha).\, g(\alpha) \bmod p(\alpha) \quad (14)$$
$$= \left(\sum_{k=0}^{2(n-1)} c_k \alpha^i \right) \bmod p(\alpha).$$

The polynomial resulting from the multiplication of $f(\alpha).\, g(\alpha)$ in (14) has coefficients

$$c_k = \sum_{i+j=k} a_i\, b_j \bmod p, \qquad (15)$$
$$0 \le i \le n-1, 0 \le j \le n-1.$$

An example of the field GF(3^2) is used to explain the basic operations in extended Galois fields. The irreducible polynomial $p(x) = x^2 + x + 2$ is used. Table 1 presents all elements of GF(3^2) as polynomials, ordered pairs of coefficients, and powers of the primitive element α.

TABLE 1 ELEMENTS OF GF(3^2) WITH PRIMITIVE POLYNOMIAL $P(X) = X^2 + X + 2$

№	Representation of GF(3^2) Elements		
	As a Polynomial	*As an Ordered Pair*	*As a Power of α*
0	0.α + 0	0 0	0
1	0.α + 1	0 1	α^0
2	0.α + 2	0 2	α^4
3	1.α + 0	1 0	α^1
4	1.α + 1	1 1	α^7
5	1.α + 2	1 2	α^6
6	2.α + 0	2 0	α^5
7	2.α + 1	2 1	α^2
8	2.α + 2	2 2	α^3

The polynomial and *n*-tuple representations are more suitable for addition and subtraction operations, while the power of the primitive element α representation allows for faster computation of multiplication and division operations. The following formulas will be used:

If $a = \alpha^x$ and $b = \alpha^y$ are two elements in GF(p^n), then their product *c* is

$$c = a.b = \alpha^x.\alpha^y = \alpha^{(x+y) \bmod (p^n-1)} \qquad (16)$$

and their quotient *d* is

$$d = \frac{a}{b} = \frac{\alpha^x}{\alpha^y} = \alpha^{(x-y) \bmod (p^n-1)}. \qquad (17)$$

As a corollary of equation (17), the multiplicative inverse element a^{-1} can be determined

$$a^{-1} = \frac{1}{a} = \frac{\alpha^{p^n-1}}{\alpha^x} = \alpha^{(p^n-1-x) \bmod (p^n-1)}. \qquad (18)$$

In Example 1 we give the examples of the arithmetic operations addition, subtraction, multiplication, and division of two elements, 2 and 8, from GF(3^2).

Example 1. Addition, subtraction, multiplication, and division of elements 2 and 8 from GF(3^2).

Addition: 2 + 8 = 2 + (2.α + 2) = 2.α + 1 = 7.

Subtraction: 2 – 8 = 2 – 2.α – 2 = 0 – 2.α = 1.α = 3.

Multiplication: $2.8 = \alpha^4.\alpha^3 = \alpha^{7 \bmod 8} = \alpha^7 = 4 = 1\ 1$.

Division: $2/8 = \alpha^4 / \alpha^3 = \alpha^{1 \bmod 8} = \alpha^1 = 3 = 1\ 0$.

Multiplicative inverse:
$2^{-1} = 1/\alpha^4 = \alpha^8/\alpha^4 = \alpha^4 = 2 = 0\ 2$.
$8^{-1} = 1/\alpha^3 = \alpha^8/\alpha^3 = \alpha^5 = 6 = 2\ 0$.

As shown in Example 1, in a field with a base other than 2, the inverse additive elements *a* and *-a* are not unique. Next, we give a specific feature 1, witch from the non-uniqueness of the inverse elements in GF(*p*), $p > 2$.

Specific Feature 1. In Galois Field GF(*p*), every nonzero element *a* has an additive inverse element $-a = p - a$.

A. Algorithm Main Steps

The pseudocode of the algorithm for generating all p-ary Reed-Solomon Codes over GF(p^n) is shown on Fig. 1.

The first step of the algorithm is to construct a Galois field GF(*p*). The second step is to construct an extended Galois field GF(p^m) by setting $p(x) = 0$, where $p(x)$ is an irreducible polynomial of degree *m* in GF(*p*). Finding the polynomial $p(x)$ of degree *m* is a computationally difficult problem. There are tables of irreducible polynomials for Galois fields of base 2 at various values of *m*. For fields with base *p* not equal to 2, sophisticated probabilistic algorithms are used to find these polynomials.

Consider the second feature of the algorithm, which involves computing the generator (2) and parity (6) polynomials for GF(p^m) with base $p > 2$. In some reference works, the subtraction operation in the formulas for computing $g(x)$ and $h(x)$ is replaced by addition due to the sameness of the addition and subtraction operations in GF(2^m).

```
algorithm p-ary Reed-Solomon Codes is
    input: a prime number p
           a natural number m
    output: all pRS codes in GF(pᵐ) and their
parameters
  CALL GFp  // Construction of a GF(p)
  CALL GFpm // Construction of a GF(pᵐ)
  n ← pᵐ - 1
  for (k = 1; k < n; k++)
  {
    CALCULATE g(x);   //Generator polynomial
    CALCULATE p(x);   //Parity polynomial
    d_min ← n - k + 1; //Hamming distance
    t ← ⌊(n - k)/2⌋;  //Number of corrected
symbols
    r ← k/n;          //Code Speed
    b = t/n;          //Code Capability
    PRINT g(x), p(x), d_min, t, r, b;
  }
```

Fig. 1. Pseudocode of the Algorithm for Generating all pRS Codes over GF(p^n).

Specific Feature 2. In formulas (2) and (6), subtraction operations must be performed due to the difference between addition and subtraction operations for fields GF(p^n) with base $p > 2$.

Example 2. Generate a ternary Reed-Solomon code over the Galois field GF(3^2) with a length of $n = 8$ and $k = 6$ information symbols.

1. Construct the GF(3) using the specified arithmetic operations of addition and multiplication:

$$c \equiv a + b \bmod 3$$
$$d \equiv a.b \bmod 3,$$

where *a*, *b*, *c*, and $d \in$ GF(3).

2. Construct an extended Galois field GF(3^2) using the primitive polynomial $p(x) = x^2 + x + 2$. To improve the speed of addition and subtraction operations in GF(3^2), elements of the field are represented as ordered pairs. For

multiplication and division operations, elements are represented as powers of the primitive element α (see Table 1).

3. Generate (8, 6) ternary RS code:

3.1. Calculate the generator polynomial (2):

$$g(x) = (x-\alpha)(x-\alpha^2) = x^2 - \alpha x - \alpha^2 x + \alpha^3 = x^2 - (\alpha+\alpha^2)x + \alpha^3 = x^2 - (10+21)x + \alpha^3 = x^2 - 01x + 22 = x^2 + 02x + 22$$

3.2. Calculate the parity polynomial (6):

$$h(x) = (x-\alpha^3)(x-\alpha^4)(x-\alpha^5)(x-\alpha^6)(x-\alpha^7)(x-1) = (x^2+\alpha^6-\alpha^2)(x-\alpha^5)(x-\alpha^6)(x-\alpha^7)(x-1) = (x^3+\alpha^3x^2+\alpha^3x+\alpha^0)(x-\alpha^6)(x-\alpha^7)(x-1) = (x^4+\alpha x^3+\alpha^6x^2+\alpha^2x+\alpha^2)(x-\alpha^7)(x-1) = (x^5+\alpha^4x^4+\alpha^7x^3+\alpha^0x^2+\alpha^7x+\alpha^5)(x-1) = \alpha^0x^6+\alpha^0x^5+\alpha^6x^4+\alpha^5x^3+\alpha^1x^2+\alpha^6x+\alpha^1 = 01x^6+01x^5+12x^4+20x^3+10x^2+12x+10$$

4. The code's characteristics are calculated:

- Minimum Hamming distance: $d_{min} = n - k + 1 = 8 - 6 + 1 = 3$;
- Maximum number of error-corrected symbols: $t = \lfloor (8-6)/2 \rfloor = 1$;
- Error Correcting Code Speed: $r = k/n = 100(6/8) = 75$ %;
- Error Correcting Code Capability: $b = t/n = 100(1/8) = 12{,}5$ %.

B. Testing the algorithm

The algorithm for generating *p*-ary Reed-Solomon codes at arbitrary prime *p* is implemented in Visual C#. It is tested for prime values *p* between 2 and 13 and all powers *m* between 2 and 8. All generated ternary RS codes over the field GF(3^2) and their parameters are given in Table 2.

Table 3 shows the representation of the elements of the field GF(5^2) as polynomials, ordered pairs and powers of the primitive element α. The field GF(5^2) is constructed using the primitive polynomial $p(x) = x^2 + x + 2$. Table 4 shows the 5RS codes over the field GF(5^2) with even $n - k$, which are more widely used because they maximize t at the corresponding k.

TABLE 2 3RS CODES OVER GF(3^2)

№	*g*(x), *h*(x)	*n*	*k*	d_{min}	*t*	*r* [%]	*b* [%]
1	$g(x) = 10 + 10x + 10x^2 + 10x^3 + 10x^4 + 10x^5 + 10x^6 + 10x^7$ $h(x) = 20 + 10x$	8	1	8	3	12,5	37,5
2	$g(x) = 02 + 20x + 01x^2 + 11x^3 + 12x^4 + 21x^5 + 10x^6$ $h(x) = 11 + 12x + 10x^2$	8	2	7	3	25	37,5
3	$g(x) = 22 + 11x + 12x^2 + 22x^3 + 12x^4 + 10x^5$ $h(x) = 01 + 01x + 21x^2 + 10x^3$	8	3	6	2	25	37,5
4	$g(x) = 12 + 20x + 12x^2 + 11x^3 + 10x^4$ $h(x) = 12 + 20x + 12x^2 + 11x^3 + 10x^4$	8	4	5	2	50	25
5	$g(x) = 12 + 11x + 01x^2 + 10x^3$ $h(x) = 12 + 22x + 22x^2 + 01x^3 + 02x^4 + 10x^5$	8	5	4	1	62,5	12,5
6	$g(x) = 22 + 20x + 10x^2$ $h(x) = 01 + 21x + 01x^2 + 02x^3 + 21x^4 + 10x^5 + 10x^6$	8	6	3	1	75	12,5
7	$g(x) = 02 + 10x$ $h(x) = 11 + 21x + 02x^2 + 20x^3 + 22x^4 + 12x^5 + 01x^6 + 10x^7$	8	7	2	0	87,5	0

It should be noted that the coefficients of the computed polynomials $g(x)$ and $p(x)$, shown in Tables 2 and 4, start from the least significant coefficient, i.e. they are written in the reverse order to those in Tables 1 and 3. The reason for this is the generalization of the algorithm to generate pRS codes at different powers *m* of the extended field GF(p^m). For example, if the GF(3^3) is used, then the element 5 ($0.\alpha^2 + 1.\alpha + 2$) of the extended Galois field is represented by the 3-tuple 210 as an output of the proposed algorithm.

TABLE 3 ELEMENTS OF GF(5^2) WITH PRIMITIVE POLYNOMIAL P(X) = X^2 + X + 2

№	As a Polynomial	As an Ordered Pair	As a Power of α	№	As a Polynomial	As an Ordered Pair	As a Power of α
0	0.α + 0	0 0	**0**	13	2.α + 3	2 3	α^{16}
1	0.α + 1	0 1	α^0	14	2.α + 4	2 4	α^{20}
2	0.α + 2	0 2	α^6	15	3.α + 0	3 0	α^{19}
3	0.α + 3	0 3	α^{18}	16	3.α + 1	3 1	α^8
4	0.α + 4	0 4	α^{12}	17	3.α + 2	3 2	α^4
5	1.α + 0	1 0	α^1	18	3.α + 3	3 3	α^{11}
6	1.α + 1	1 1	α^{17}	19	3.α + 4	3 4	α^9
7	1.α + 2	1 2	α^{14}	20	4.α + 0	4 0	α^{13}
8	1.α + 3	1 3	α^{15}	21	4.α + 1	4 1	α^{22}
9	1.α + 4	1 4	α^{10}	22	4.α + 2	4 2	α^3
10	2.α + 0	2 0	α^7	23	4.α + 3	4 3	α^2
11	2.α + 1	2 1	α^{21}	24	4.α + 4	4 4	α^5
12	2.α + 2	2 2	α^{23}				

TABLE 4 5RS CODES OVER GF(5^2)

№	*g*(x), *h*(x)	*n*	*k*	d_{min}	*t*	*r* [%]	*b* [%]
1	$g(x) = 04 + 20x + 01x2 + 33x^3 + 44x^4 + 24x^5 + 22x^6 + 14x7 + 21x^8 + 30x^9 + 02x^{10} + 12x^{11} + 13x^{12} + 42x^{13} + 11x^{14} + 34x^{15} + 23x^{16} + 43x^{17} + 40x^{18} + 03x^{19} + 41x^{20} + 32x^{21} + 10x^{22}$ $h(x) = 22 + 23x + 10x^2$	24	2	23	11	8,33	45,83
2	$g(x) = 30 + 32x + 31x^2 + 11x^3 + 44x^4 + 14x^5 + 43x^6 + 23x^7 + 04x^8 + 42x^9 + 41x^{10} + 22x^{11} + 03x^{12} + 04x^{13} + 12x^{14} + 04x^{15} + 22x^{16} + 21x^{17} + 31x^{18} + 03x^{19} + 10x^{20}$ $h(x) = 30 + 23x + 40x^2 + 02x^3 + 10x^4$	24	4	21	10	16,66	41,66
3	$g(x) = 24 + 11x + 41x^2 + 02x^3 + 41x^4 + 22x^5 + 24x^6 + 01x^7 + 33x^8 + 11x^9 + 32x^{10} + 33x^{11} + 30x^{12} + 03x^{13} + 33x^{14} + 04x^{15} + 12x^{16} + 43x^{17} + 10x^{18}$ $h(x) = 43 + 33x + 22x^2 + 22x^3 + 23x^4 + 12x^5 + 10x^6$	24	6	19	9	25	37,5
4	$g(x) = 32 + 01x + 43x^2 + 34x^3 + 14x^4 + 21x^5 + 03x^6 + 32x^7 + 33x^8 + 22x^9 + 43x^{10} + 33x^{11} + 34x^{12} + 04x^{13} + 24x^{14} + 34x^{15} + 10x^{16}$ $42 + 11x + 22x^2 + 31x^3 + 10x^4 + 14x^5 + 04x^6 + 21x^7 + 10x^8$	24	8	17	8	33,33	33,33
5	$g(x) = 43 + 21x + 31x^2 + 33x^3 + 13x^4$	24	10	15	7	41,	29,

№	$g(x)$, $h(x)$	n	k	d_{min}	t	r [%]	b [%]
	$+ 22x^5 + 04x^6 + 32x^7 + 14x^8 + 11x^9 + 33x^{10} + 22x^{11} + 40x^{12} + 42x^{13} + 10x^{14}$ $h(x) = 24 + 42x + 11x^2 + 23x^3 + 02x^4 + 12x^5 + 32x^6 + 14x^7 + 42x^8 + 13x^9 + 10x^{10}$					66	16
6	$g(x) = 20 + 21x + 32x^2 + 33x^3 + 11x^4 + 21x^5 + 31x^6 + 01x^7 + 31x^8 + 42x^9 + 40x^{10} + 12x^{11} + 10x^{12}$ $h(x) = 20 + 34x + 32x^2 + 22x^3 + 11x^4 + 34x^5 + 31x^6 + 04x^7 + 31x^8 + 13x^9 + 40x^{10} + 43x^{11} + 10x^{12}$	24	12	13	6	50	25
7	$g(x) = 02 + 21x + 01x^2 + 34x^3 + 03x^4 + 22x^5 + 13x^6 + 23x^7 + 32x^8 + 30x^9 + 10x^{10}$ $h(x) = 44 + 10x + 31x^2 + 01x^3 + 40x^4 + 11x5 + 12x^6 + 21x^7 + 41x^8 + 03x^9 + 24x^{10} + 44x^{11} + 13x^{12} + 20x^{13} + 10x^{14}$	24	14	11	5	58, 33	20, 83
8	$g(x) = 40 + 01x + 22x^2 + 02x^3 + 13x^4 + 14x^5 + 44x^6 + 14x^7 + 10x^8$ $h(x) = 10 + 01x + 01x^2 + 41x^3 + 14x^4 + 20x^5 + 24x^6 + 32x^7 + 24x^8 + 02x^9 + 43x^{10} + 24x^{11} + 41x^{12} + 04x^{13} + 03x^{14} + 41x^{15} + 10x^{16}$	24	16	9	4	66, 66	16, 66
9	$g(x) = 12 + 11x + 22x^2 + 11x^3 + 32x^4 + 24x^5 + 10x^6$ $h(x) = 31 + 22x + 41x^2 + 01x^3 + 14x^4 + 44x^5 + 24x^6 + 03x^7 + 22x^8 + 22x^9 + 32x^{10} + 44x^{11} + 20x^{12} + 01x^{13} + 33x^{14} + 02x^{15} + 43x^{16} + 31x^{17} + 10x^{18}$	24	18	7	3	75	12,5
10	$g(x) = 41 + 32x + 42x^2 + 33x^3 + 10x^4$ $h(x) = 34 + 42x + 31x^2 + 04x^3 + 12x^4 + 41x^5 + 11x^6 + 13x^7 + 04x^8 + 32x^9 + 34x^{10} + 33x^{11} + 24x^{12} + 11x^{13} + 12x^{14} + 43x^{15} + 31x^{16} + 34x^{17} + 22x^{18} + 22x^{19} + 10x^{20}$	24	20	5	2	83, 33	8, 33
11	$g(x) = 24 + 20x + 10x^2$ $h(x) = 43 + 10x + 11x^2 + 01x^3 + 11x^4 + 04x^5 + 02x^6 + 23x^7 + 30x^8 + 42x^9 + 02x^{10} + 03x^{11} + 23x^{12} + 21x^{13} + 43x^{14} + 32x^{15} + 32x^{16} + 03x^{17} + 42x^{18} + 01x^{19} + 21x^{20} + 30x^{21} + 10x^{22}$	24	22	3	1	91, 66	4,16

CONCLUSIONS

The article proposes an algorithm for generating p-ary Reed-Solomon codes and identifies two specific features of an algorithm for an arbitrary prime p, greater than 2.

In the developed algorithm for generating p-ary Reed-Solomon codes, the most tedious problem is that of constructing an extension of a Galois field GF(p^m) at an arbitrary prime p. As p and m grow, the time to find a primitive polynomial $p(x)$ with which to generate the extended field grows exponentially. This also increases the memory required to store the powers of the primitive element α.

When selecting a suitable RS code, the theoretical and applied aspects of its characteristics must be considered. From a theoretical point of view, the codes are compared in terms of their ability to maximize the channel capacity, which automatically leads to a requirement for a high RS code rate, the signal-to-noise ratio and the number of errors that the code can correct and detect. From an application point of view, the complexity of the implementation, the processing delay, and the evaluation of the possibility of retransmitting the data packet if the errors cannot be corrected are considered. In this respect, the use of a prime number p greater than two leads to a reduction in the length of the RS code used to transmit the same amount of information while maintaining the code's ability to correct.

To summarise, the proposed algorithm has a higher time and memory complexity in its first two steps, which are only executed once, compared to the original RS algorithm that operates at base 2. The algorithm's main advantage is that, by using a prime p greater than 2, it significantly reduces the length of the pRS code while maintaining the same error-correcting capability as the RS code with base 2.

As future work, a speed comparison could be conducted on concrete computational systems between the proposed algorithm running on an arbitrary prime p and the original algorithm running with prime 2.

ACKNOWLEDGMENTS:

The article was prepared with the financial support of the National Scientific Program “Security and Defence”, funded by the Ministry of Education and Science of the Republic of Bulgaria, in implementation of Decision № 731 of 21.10.2021 of the Council of Ministers of the Republic of Bulgaria.

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

REFERENCES

[1] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” Journal of the Society for Industrial & Applied Mathematics 8, No. 2, pp. 300-304, 1960.

[2] H. Hoeve, J. Timmermans and L. Vries, “3.4 Error Correction and Concealment in Compact Disc Systems, ” in Origins and Successors of the Compact Disc, Contributions of Philips to Optical Storage: Springer Link, 2009, pp. 82.

[3] J. D. Key, **“**Some error-correcting codes and their applications, **”** in Applied Mathematical Modeling: A Multidisciplinary Approach, Chapman & Hall/CRC Press, 1999.

[4] H. Chang, C. Shung and C. Lee, “A Reed–Solomon Product-Code (RS-PC) Decoder Chip for DVD Applications, ” IEEE Journal of Solid-State Circuits, Vol. 36, No. 2, pp. 229-238, February 2001.

[5] X. Liu, H. Jia and C. Ma. “Error-Correction codes For Optical Disc Storage, ” in Advances in Optical Data Storage Technology, Proceedings of SPIE Vol. 5643, pp. 342-347, 2005.

[6] J. A. Lin and C. S. Fuh, “2D Barcode Image Decoding, ” in Mathematical Problems in Engineering, Article ID 848276, 10 pages, 2013. https://doi.org/10.1155/2013/848276 .

[7] A. J. McAuley, “Reliable broadband communication using a burst erasure correcting code. ” in Proceedings of the ACM symposium on Communications architectures & protocols, pp. 297-306, ACM, 1990, https://dl.acm.org/doi/pdf/10.1145/99508.99566 .

[8] H. -C. Lee, J. -H. Wu, C. -H. Wang and Y. -L. Ueng, "A Graph-Based Soft-Decision Decoding Scheme for Reed-Solomon Codes," in IEEE Journal on Selected Areas in Information Theory, vol. 4, pp. 420-433, 2023, https://doi.org/10.1109/JSAIT.2023.3315453 .

[9] Y.Chen, “Thermal Management and Data Archiving in Data Centers,” Ph.D. thesis, Auburn University, Auburn, Alabama, 2016.

[10] T. N. Hewage, M. N. Halgamuge, A. Syed, and G. Ekici, “Big data techniques of Google, Amazon, Facebook and Twitter. ” in Journal of Communications Vol. 13, No. 2, pp. 94-100, February 2018.

[11] A. Chiniah and A.Mungur, “On the Adoption of Erasure Code for Cloud Storage by Major Distributed Storage Systems,” in EAI Endorsed Transactions on Cloud Systems, 7(21), e1-e11, 2022.

[12] H.-U. Kim and J.-K. Kang, “High-speed Serial Interface using PWAM Signaling Scheme,” in 19th International SoC Design

Conference (ISOCC), Gangneungsi, Korea, pp. 255-256, 2022, https://doi.org/10.1109/ISOCC56007.2022.10031330 .

[13] N. Stojanović, C. Prodaniuc, Z. Liang, J. Wei, S. Calabró, T. Rahman and C. Xie, "4D PAM-7 Trellis Coded Modulation for Data Centers," in IEEE Photonics Technology Letters, Vol. 31, No. 5, pp. 369-372, 1 March 2019, https://doi.org/10.1109/LPT.2019.2895686 .

[14] K. Matheus and T. Königseder, Automotive Ethernet. Cambridge University Press, 2021.

[15] S. Nabipour and M. Gholizade, Arithmetic Operators over Finite Field GF (2^m) in BCH and Reed-Solomon Codes, arXiv preprint arXiv:2310.12319. 2023, [Online]. Available: https://arxiv.org/ftp/arxiv/papers/2310/2310.12319.pdf. [Accessed: Jan. 7, 2024].

[16] R. C. Bose and D.K. Ray-Chaudhuri. "On a class of error correcting binary group codes," in Information and Control, Volume 3, Issue 1, pp. 68–79, March 1960.

[17] N.Atti, G. Diaz–Toca and H. Lombardi, The Berlekamp-Massey Algorithm revisited, in Applicable Algebra in Engineering, Communication and Computing 17(1), pp. 75–82, 2006, https://doi.org/10.1007/s00200-005-0190-z .

[18] J. A. M.Naranjo, J. A. López-Ramos and L. G. Casado, "Applications of the extended Euclidean algorithm to privacy and secure communications, " in Proceedings of the 10th International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE 2010, pp. 27–30, June 2010.

[19] A. Beletsky, "An Effective Algorithm for the Synthesis of Irreducible Polynomials over a Galois Fields of Arbitrary Characteristics," in WSEAS Transactions on Mathematics 20, pp. 508-519, 2021.