Computing and Informatics, Vol. 39, 2020, 1311–1335, doi: 10.31577/cai_2020_6_1311

DECODING FIVE TIMES EXTENDED REED SOLOMON CODES USING SYNDROMES

Peter Farkaš

Institute of Multimedia ICT, FEI STU
Ilkovičova 3
812 19 Bratislava, Slovakia
&
Institute of Applied Informatics
Faculty of Informatics Pan-European University, Slovakia
Tematínska 10
851 05 Bratislava, Slovakia
e-mail: p.farkas@ieee.org

Martin Rakús

Institute of Multimedia ICT, FEI STU
Ilkovičova 3
812 19 Bratislava, Slovakia
e-mail: martin.rakus@stuba.sk

Abstract. Recently a new family of five times extended Reed Solomon codes constructed over certain finite fields $GF(2^{\zeta})$, where $\zeta \geq 3$ is an odd integer, was discovered. Until now only an erasure decoding algorithm for these codes was published. In this paper a new decoding algorithm is presented, which allows correcting up to two errors in a codeword from the five times extended Reed Solomon codes. The proposed decoding algorithm is based on syndrome usage.

Keywords: Algorithm, extended Reed Solomon codes, error correction decoding, syndromes

Mathematics Subject Classification 2010: 58F15, 58F17, 53C35

1 INTRODUCTION

Reed Solomon (RS) codes were discovered more than 60 years ago [3]. Their practical importance was proven by mass applications in such systems as CD or DVB [4]. More surprisingly, in recent years a new interest in these codes has risen in the research community. It is caused by the fact that they are extensively used in cloud technology, namely in distributed storage systems [5, 6, 7, 8, 9]. In [1] it was shown that RS codes could be extended not only three times, but even five times when they are constructed over certain finite fields. For such five times extended RS codes a decoding algorithm is known [2] which is suitable only for erasure decoding. In case of erasures the positions are known in a received vector and therefore only the erasure values have to be found out during decoding. In contrast to this, the positions of the errors in received vectors are unknown; therefore both error values and error positions have to be determined during decoding. The error correction is useful in many applications of RS codes [4]. For example, in [10] syndrome decoding is patented for random access-based computer memory systems by IBM Corp. Interestingly, it contains a reference to [11] in which the authors of this paper proposed a related code which inspired the construction of five times extended Reed Solomon codes [1].

In this paper a decoding algorithm is presented which allows correcting at least 2 errors in five times extended RS codes from [1].

The paper is organized as follows. In Section 2 a short introduction to the original (not extended) RS codes error correcting decoding via syndromes is presented. In Section 3 the error correcting decoding of the original RS codes is given. In Section 4 the novel error correcting algorithm for five times extended RS codes introduced in [1] is presented. In Section 5 some remarks on the decoding implementation are given. In Section 6 the complexity estimation of the algorithm is made. Conclusions in Section 7 summarize the results of the paper.

2 A SHORT INTRODUCTION TO ORIGINAL RS CODES

RS codes are linear block codes which could be described by many mathematical tools and which have many interesting connections with different mathematical branches. In this section only a short introduction to RS codes is given. The interested reader could find a more detailed explanation of RS codes and their applications for example in [4]. To explain the proposed decoding algorithm a basic description of RS codes as linear block codes via matrices and as cyclic codes using polynomials is given.

A linear block code is defined as a k-dimensional subspace of an n-dimensional vector space over a finite field $GF(q)$. Basic parameters of linear block codes are: the codeword length n, the number of information symbols k in each codeword and the code distance d_m which is a minimal Hamming distance between any two codewords. Sometimes linear block codes are in shorthand using a triple $[n, k, d_m]$,

which contains these fundamental parameters. The code distance d_m and the number of correctable errors t in a linear code are connected by the following inequality:

$$2t + 1 \leq d_m.$$

Because a linear block code is defined as a k-dimensional subspace of an n-dimensional vector space over a finite field $GF(q)$ it could be described via a $k \times n$ generator matrix $\mathbf{G}_{k\times n}$ which contains as rows k linearly independent vectors with length n. In systematic form:

$$\mathbf{G}_{k\times n} = \left[\mathbf{I}_{k\times k} \mid \mathbf{P}_{k\times(n-k)}\right] \tag{1}$$

where $\mathbf{I}_{k\times k}$ and $\mathbf{P}_{k\times(n-k)}$ are the identity and parity matrices, respectively. For decoding purposes a control matrix is defined as:

$$\mathbf{H}_{(n-k)\times n} = \left[\mathbf{P}^T_{(n-k)\times k} \mid \mathbf{I}_{(n-k)\times(n-k)}\right] \tag{2}$$

where $\mathbf{I}_{(n-k)\times(n-k)}$ and $\mathbf{P}^T_{(n-k)\times k}$ are the identity and transposed parity matrices, respectively. The following equation is valid:

$$\mathbf{G}_{k\times n}.\mathbf{H}^T_{n\times(n-k)} = \mathbf{0}_{k\times(n-k)} \tag{3}$$

where $\mathbf{H}^T_{n\times(n-k)}$ and $\mathbf{0}_{k\times(n-k)}$ are the transposed matrix $\mathbf{H}$ and all zeros matrix, respectively.

In [1] a new family of codes constructed over $GF(2^\zeta)$, where $\zeta \geq 3$ is an odd integer, using the $\mathbf{H}$ matrix (4) was proposed.

$$\mathbf{H} = \begin{bmatrix} \alpha^0 & \alpha^0 & \cdots & \alpha^0 & \alpha^0 & \alpha^0 & 1 & 0 & 0 & 0 & 0 \\ \alpha^{(q-2)} & \alpha^{(q-3)} & \cdots & \alpha^2 & \alpha^1 & \alpha^0 & 0 & 1 & 0 & 0 & 0 \\ \alpha^{2(q-2)} & \alpha^{2(q-3)} & \cdots & \alpha^4 & \alpha^2 & \alpha^0 & 0 & 0 & 1 & 0 & 0 \\ \alpha^{3(q-2)} & \alpha^{3(q-3)} & \cdots & \alpha^6 & \alpha^3 & \alpha^0 & 0 & 0 & 0 & 1 & 0 \\ \alpha^{4(q-2)} & \alpha^{4(q-3)} & \cdots & \alpha^8 & \alpha^4 & \alpha^0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

The basic parameters of this family of block codes can be characterized by a triple $[q + 4, q - 1, 5]$.

3 ERROR CORRECTING DECODING OF ORIGINAL RS CODES

The definition of original RS codes correcting t errors requires that each codeword of these cyclic codes has $2t$ consecutive and distinct elements of a finite field as roots. In other words the generating polynomial should also have the same property. In $GF(2^\zeta)$, $\zeta \geq 2$ we get:

$$g(x) = \left(x + \alpha^j\right)\left(x + \alpha^{j+1}\right)\left(x + \alpha^{j+2}\right)\ldots\left(x + \alpha^{j+2t-1}\right) \tag{5}$$

where j is an integer (for convenience usually equal to 0 or 1). In the following explanation we will use $j = 0$ for simplicity. Then the generator polynomial can be

written as follows:

$$g(x) = \left(x + \alpha^0\right)\left(x + \alpha^1\right)\left(x + \alpha^2\right)\ldots\left(x + \alpha^{2t-1}\right). \tag{6}$$

The original RS codes could be described using the following control matrix:

$$\mathbf{H}_{RS} = \begin{bmatrix} \alpha^0 & \alpha^0 & \cdots & \alpha^0 & \alpha^0 & \alpha^0 \\ \alpha^{(q-2)} & \alpha^{(q-3)} & \cdots & \alpha^2 & \alpha^1 & \alpha^0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ \alpha^{(2t-1)(q-2)} & \alpha^{(2t-1)(q-3)} & \cdots & \alpha^{(2t-1)2} & \alpha^{(2t-1)} & \alpha^0 \end{bmatrix} \tag{7}$$

which is in fact a Vandermonde matrix $\mathbf{V}_{2t\times n}$ over $GF(2^\zeta)$.

There are numerous different decoding algorithms for original RS codes. In this section, we will describe only the basic algorithm for error correction of codewords of RS codes based on syndromes. The reason is that this algorithm is the most related to the proposed decoding algorithm for the five times extended RS codes, which will be described later.

The main goal of error correcting codes and their decoding is to decrease the number of errors and/or erasures which can occur during the information transmission or storage. Next, we will focus on error correction. The codewords of RS codes can be described as vectors in which the coordinates are symbols from the underlying finite field:

$$\mathbf{c} = (c_{n-1}, c_{n-2}, \ldots, c_1, c_0). \tag{8}$$

Another possibility is to use polynomials for descriptions:

$$c(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \ldots + c_1x^1 + c_0x^0. \tag{9}$$

The symbols c_i in (8) and 9 are elements from $GF(q)$. In the present digital era the most practical choice are finite fields $GF(2^\zeta)$, which are extensions of the binary finite field. $\zeta \geq 2$ is positive integer.

In such fields the i^{th} error in a codeword could be described using two unknowns – the so called error value $Y_i \in GF(2^\zeta)$ and its position, which is given by a corresponding error locator $X_i \in GF(2^\zeta)$. The decoder needs to calculate both of these values for each error in order to correct one error in a codeword. On the other hand, each polynomial 9) representing a codeword from the RS code has $2t$ roots which are consecutive elements in $GF(2^\zeta)$. This allows the formation of $2t$ equations in order to correct t errors in one codeword from the original RS code. The following explanation and example of the basic decoding algorithm makes it more obvious.

The model of an additive error channel is illustrated in Figure 1.

Because for any codeword of a cyclic code the following two equations hold:

$$c(x)\,\mathtt{mod}\,g(x) = 0 \tag{10}$$

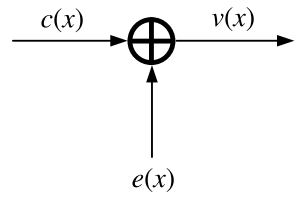

Figure 1. Additive error channel model. $\oplus$ denotes a polynomial addition over $GF(2^{\zeta})$, $c(x)$ is the transmitted codeword, $e(x)$ is an error polynomial and $v(x)$ is the received polynomial which can contain errors.

and

$$g(\alpha^k) = 0; \quad k = j, (j+1), \ldots, (j+2t-1) \tag{11}$$

then the following equation is also valid:

$$c(\alpha^k) = 0; \quad k = j, (j+1), \ldots, (j+2t-1). \tag{12}$$

One of the many algorithms which are known for error correcting decoding for original RS codes is the following syndrome method. The first step consists of inserting the roots, which define the concrete RS code via (5) into the received polynomial $v(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + \ldots + v_1x^1 + v_0x^0$:

$$S_k = v(\alpha^k) = c(\alpha^k) + e(\alpha^k); \quad k = j, (j+1), \ldots, (j+2t-1). \tag{13}$$

If $j = 0$:

$$\begin{aligned} S_0 &= v(\alpha^0), \\ S_1 &= v(\alpha^1), \\ &\vdots \\ S_{2t-1} &= v(\alpha^{2t-1}) \end{aligned} \tag{14}$$

where S_j denotes the j^{th} syndrome, $j = 0, 1, \ldots, 2t-1$. From (12) and (13):

$$S_k = e(\alpha^k); \quad k = j, (j+1), \ldots, (j+2t-1). \tag{15}$$

Therefore each syndrome is dependent only on the error polynomial written as $e(x) = e_{n-1}x^{n-1} + e_{n-2}x^{n-2} + \ldots + e_1x^1 + e_0x^0$ and it could be expressed also via the unknowns which have to be calculated in order to correct the errors. Namely the error locators X_i; $i = 0, 1, 2, \ldots, t$ and error values Y_i; $i = 0, 1, 2, \ldots, t$:

$$S_k = \sum_{i=1}^{t} Y_i X_i^k; \quad k = 0, 1, \ldots, 2t-1. \tag{16}$$

(At this stage it is not known how many errors τ occurred in reality, but we suppose that $\tau \leq t$ and so we start with the worst case assumption, that $\tau = t$.)

Actually (16) is a system of nonlinear equations which could not be solved directly. It could be better seen after writing (16) in a more detailed fashion:

$$\begin{array}{ccccccccc} Y_1 & + & Y_2 & \cdots & + & Y_t & = & S_0, \\ Y_1X_1 & + & Y_2X_2 & \cdots & + & Y_tX_t & = & S_1, \\ Y_1X_1^2 & + & Y_2X_2^2 & \cdots & + & Y_tX_t^2 & = & S_2, \\ & & & & \vdots & & & \\ Y_1X_1^{2t-1} & + & Y_2X_2^{2t-1} & \cdots & + & Y_tX_t^{2t-1} & = & S_{2t-1}. \end{array} \tag{17}$$

A clever way to deal with this difficulty is to first find the error locators and then after introducing them into (16) the system turns into a system of linear equations which could be solved by standard algorithms, for example Gauss elimination.

The error locator polynomial could be used for this purpose. The error locator polynomial: $\lambda(x) = \lambda_t x^t + \lambda_{t-1}x^{t-1} + \ldots + \lambda_1 x + 1$ is a polynomial which has roots, which are error locators. It could be expressed as follows:

$$\lambda(X_i) = 0 \quad i = 0, 1, \ldots, t. \tag{18}$$

In order to find an error locator polynomial we can rewrite (18):

$$\lambda_t X_i^t + \lambda_{t-1}X_i^{t-1} + \ldots + \lambda_1 X_i + 1 = 0 \quad i = 0, 1, \ldots, t. \tag{19}$$

(19) could be multiplied by $Y_iX_i^z$; $i = 1, 2, \ldots, t$; $z \in Z$:

$$\lambda_t Y_i X_i^{t+z} + \lambda_{t-1}Y_iX_i^{t+z-1} + \ldots + \lambda_1 Y_iX_i^z + Y_iX_i^z = 0 \quad i = 0, 1, \ldots, t. \tag{20}$$

Now we can write (18) for fixed values of $z \in Z$. For $z = 0$ we get:

$$\begin{array}{ccccccccc} Y_1X_1^0 & + & \lambda_1Y_1X_1^1 & + & \cdots & + & \lambda_tY_1X_1^t & = & 0, \\ Y_2X_2^0 & + & \lambda_1Y_2X_2^1 & + & \cdots & + & \lambda_tY_2X_2^t & = & 0, \\ & & & & & & \vdots & & \\ Y_tX_t^0 & + & \lambda_1Y_tX_t^1 & + & \cdots & + & \lambda_tY_tX_t^t & = & 0. \end{array} \tag{21}$$

Summation of (21) gives us:

$$S_0 + \lambda_1 S_1 + \ldots + \lambda_t S_t = 0. \tag{22}$$

Similarly for $z = 1$ we get:

$$\begin{array}{ccccccccc} Y_1X_1^1 & + & \lambda_1Y_1X_1^2 & + & \cdots & + & \lambda_tY_1X_1^{t+1} & = & 0, \\ Y_2X_2^1 & + & \lambda_1Y_2X_2^2 & + & \cdots & + & \lambda_tY_2X_2^{t+1} & = & 0, \\ & & & & & & \vdots & & \\ Y_tX_t^1 & + & \lambda_1Y_tX_t^2 & + & \cdots & + & \lambda_tY_tX_t^{t+1} & = & 0. \end{array} \tag{23}$$

Summation of (23) gives us:

$$S_1 + \lambda_1 S_2 + \ldots + \lambda_t S_{t+1} = 0. \tag{24}$$

We can continue in a similar way until we get the following system of equations:

$$\begin{array}{ccccccccc} S_0 & + & \lambda_1 S_1 & + & \cdots & + & \lambda_t S_t & = & 0, \\ S_1 & + & \lambda_1 S_2 & + & \cdots & + & \lambda_t S_{t+1} & = & 0, \\ & & & & & & \vdots & & \\ S_{t-1} & + & \lambda_1 S_t & + & \cdots & + & \lambda_t S_{2t-1} & = & 0. \end{array} \tag{25}$$

This system of linear equations could be solved by any standard method, for example by Gauss-Jordan elimination. As a result we get the coefficients of the error locator polynomial.

The error locator polynomial determination from the calculated syndromes could also be done using different approaches, for example the Berlekamp-Massey algorithm [12, 13].

In the third step the locators are found via Chien search [14]. It is, in principle, a slightly augmented brute force algorithm in which the symbols are inserted consecutively into the locator polynomial, evaluated and tested if:

$$\lambda(X_i) = 0 \quad i = 0, 1, \ldots, \tau; \quad \tau \leq t \tag{26}$$

until we get τ locators. This algorithm is possible because the underlying field is finite. It will stop when we get enough solutions. In other words it means that with high probability we do not usually have to insert all nonzero elements.

Nevertheless, the Chien search has high complexity. If the error locator has a small degree it could be solved directly [15, 16, 17]. Or theorem 3.2.15 in [18] can be used with much smaller computational complexity. For example the method in [16] has very low complexity.

The result of this step will be the set of error locators: X_i; $i = 0, 1, \ldots, \tau$, where τ is the number of errors which actually occurred during transmission in the decoded codeword.

The fourth step is the calculation of error values. This can be done thanks to known syndromes, which could be expressed in another form:

$$S_k = \sum_{i=1}^{\tau} Y_i X_i^k \quad k = 0, 1, \ldots, 2t-1. \tag{27}$$

In this step it remains to calculate the error values Y_i; $i = 0, 1, \ldots, \tau$. This could be accomplished simply by solving the set of equation given by (27). At first we have to insert the values of error locators into it which will transform it into a set of linear equations. (The error locator values are elements of the finite field and

therefore also their powers are elements of the same finite field.) Then this set of linear equations could be solved by the Sarus rule or by Gauss-Jordan elimination.

In this step the attempt to correct the received word could be done by adding the error values to received symbols in positions determined by error locators.

One may ask how the actual number of errors $\tau \leq t$ could be obtained in the third step of the above algorithm. When using Gauss-Jordan elimination it is quite straightforward. When using the Sarus rule we have to test if the determinant of the system is zero. We have to suppose that $\tau = t$ first. If the determinant of the matrix corresponding to the system of equations is zero, we will have to suppose that $\tau = t - 1$ and again compute the new determinant and test if it is zero and if not then we can conclude that $\tau = t - 1$. If it is zero we can continue to decrease τ and modify the corresponding system matrix by deleting the last row and last column until we get a nonzero determinant. It has to be said that there exists a much more efficient method to find the error locator, namely the Berlecamp Massey algorithm. The details can be found in [12, 13].

The last step is obtaining the estimation of the transmitted or stored codeword from the received word – the actual error correction of the received word. This is done by adding the error values to the positions determined by error locators in the received codeword.

4 A DECODING ALGORITHM FOR ERROR CORRECTION FOR FIVE TIMES EXTENDED RS CODES

In [1] it was proven that the code distance of each code from this family is 5. This code distance potentially allows for correcting up to two errors in a codeword. This is a necessary but insufficient condition. The additional condition which has to be fulfilled is the knowledge of a decoding algorithm.

Unfortunately the approach presented in the previous section is not applicable to codes proposed in [1]. It is because (4) contains not only a Vandermonde matrix as a submatrix, but also an additional identity matrix in juxtaposition. It becomes obvious by inspection of the matrix (4), which can be represented in a compact form as:

$$\mathbf{H}_{5\times n} = \left[\mathbf{V}_{5\times(q-1)} \mid \mathbf{I}_{5\times 5}\right]. \tag{28}$$

Therefore in this section a new specialized error correcting algorithm will be presented for codes from [1]. The main problem is that the classical syndrome method in this case is not able to distinguish between different error patterns by analyzing the values of syndromes. The approach which will overcome this difficulty is similar to using sieves for mechanical separation by size. In other words the algorithm deals first with error patterns detectable by syndromes. The rest of the error patterns are processed at the end of decoding.

At first we will introduce notation which will allow us to underline some properties, which will be exploited in the decoding algorithm. The codewords of the codes

from [1] will be denoted as follows:

$$\mathbf{c} = (c_{k-1}, c_{k-2}, \ldots, c_1, c_0, p_0, p_1, p_2, p_3, p_4) \tag{29}$$

where the left part contains information vector: $\mathbf{c} = (m_{k-1}, m_{k-2}, \ldots, m_1, m_0)$, or in other words:

$$c_i = m_i \quad i = 0, 1, \ldots, k-1 \tag{30}$$

and

$$p_I = \sum_{i=0}^{k-1} m_i \alpha^I; \quad I = 0, 1, \ldots, 4. \tag{31}$$

It is obvious that (30) and (31) could also be used for systematic encoding of codes from [1]. The received vector which has to be decoded and therefore can contain errors will be denoted as follows:

$$\mathbf{v} = (v_{k-1}, v_{k-2}, \ldots, v_1, v_0, r_0, r_1, r_2, r_3, r_4). \tag{32}$$

The decoding algorithm starts with the calculation of 5 syndromes via multiplying the received vector by the transposed control matrix (4):

$$\mathbf{s} = \mathbf{v}.\mathbf{H}_{5\times n}^T. \tag{33}$$

The resulting vector will contain, as its coordinates, the desired syndromes:

$$\mathbf{s} = (S_0, S_1, S_2, S_3, S_4). \tag{34}$$

The second step consists of analysing $\mathbf{s}$. The algorithm will continue in depending on this analysis.

1. If all coordinates of $\mathbf{s}$ are zeros:

   $$\mathbf{s} = (0, 0, 0, 0, 0) \tag{35}$$

   the algorithm ends, and it is supposed that no errors occurred, therefore:

   $$\hat{\mathbf{c}} = \mathbf{v} \tag{36}$$

   where $\hat{\mathbf{c}}$ is the estimation of the transmitted or stored codeword.

2. If only one syndrome S_I (one coordinate) is nonzero and all other four are zeros, then it is supposed that only one error occurred in the parity symbol corresponding to the position of the nonzero element in $\mathbf{s}$. This symbol can be corrected simply by calculating the corresponding parity symbol again using the received non corrupted symbols corresponding to information symbols. Assuming that $S_I \neq 0$ then $S_I = Y$ and:

   $$\hat{\mathbf{p}}_I = S_I + r_I. \tag{37}$$

   For the other coordinates of the decoded word (36) is valid.

3. If only two syndromes e.g. $S_I \neq 0$ and $S_J \neq 0$ and all other are zeros, then it is supposed that only two errors occurred and only in the parity positions. Otherwise, if one error is in the parity part and the other in the information part or both errors are in the information part, or only one error occurred in the information part then at least one other syndrome would be nonzero. This follows from the properties of Vandermonde matrices. In this case the two corresponding parity symbols could be computed similarly as in case 2. Now $S_I \neq 0 \Longrightarrow S_I = Y_1$ and $S_J \neq 0 \Longrightarrow S_J = Y_2$ then:

$$\begin{aligned} \hat{\mathbf{p}}_I &= S_I + r_I, \\ \hat{\mathbf{p}}_J &= S_J + r_J. \end{aligned} \tag{38}$$

Note: In the following steps a situation when one or two errors can occur in the information part, or one error can occur in the information part and one in the parity part, has to be investigated and decided. For these tests a set of auxiliary variables is computed:

$$\begin{aligned} A_1 &= \tfrac{S_1}{S_0}, \\ A_2 &= \tfrac{S_2}{S_1}, \\ A_3 &= \tfrac{S_3}{S_2}, \\ A_4 &= \tfrac{S_4}{S_3}. \end{aligned} \tag{39}$$

In connection with (39) possible division by 0 has to be taken into account in any practical implementation e.g. by substituting a value not present in the given $GF(q)$.

4. At first let us assume that one error occurred in the information part. In this case all of the following equations must be fulfilled:

$$\begin{aligned} S_0 &= Y, \\ S_1 &= XY, \\ S_2 &= X^2Y, \\ S_3 &= X^3Y, \\ S_4 &= X^4Y. \end{aligned} \tag{40}$$

From (40) follows that:

$$A_1 = A_2 = A_3 = A_4 = X = \alpha^j \tag{41}$$

where j gives the position of the error in the information part to which the error value $Y = S_0$ (from (40)) has to be added in order to correct the received word:

$$\hat{c}_j = v_j + Y. \tag{42}$$

5. If in one or in two cases $A_i \neq A_{i+1}$, $i = 1, 2, 3$ then one error occurred in the information part and one in the parity part. In order to test in which position in the parity part the error occurred, one of the following 5 sets of equations is valid. In order to determine the error position in the parity part one of the following tests has to hold.

   (a) If $J = 0$ then:

$$\begin{aligned} S_0 &= Y_1 + Y_2, \\ S_1 &= X_1 Y_1, \\ S_2 &= X_1^2 Y_1, \\ S_3 &= X_1^3 Y_1, \\ S_4 &= X_1^4 Y_1. \end{aligned} \tag{43}$$

   Test if the following is true:

$$A_1 \neq A_2 \wedge A_2 = A_3 \wedge A_3 = A_4. \tag{44}$$

   From (43) follows that:

$$\begin{aligned} X_1 &= \tfrac{S_2}{S_1} = \alpha^j, \\ X_2 &= \alpha^0, \\ Y_1 &= \tfrac{S_1^2}{S_2}, \\ Y_2 &= S_0 + \tfrac{S_1^2}{S_2}. \end{aligned} \tag{45}$$

   (b) If $J = 1$ then:

$$\begin{aligned} S_0 &= Y_1, \\ S_1 &= X_1 Y_1 + Y_2, \\ S_2 &= X_1^2 Y_1, \\ S_3 &= X_1^3 Y_1, \\ S_4 &= X_1^4 Y_1. \end{aligned} \tag{46}$$

   Test if the following is true:

$$A_1 \neq A_2 \wedge A_2 \neq A_3 \wedge A_3 = A_4. \tag{47}$$

   From (46) follows that:

$$\begin{aligned} X_1 &= \tfrac{S_3}{S_2} = \alpha^j, \\ X_2 &= \alpha^1, \\ Y_1 &= S_0, \\ Y_2 &= S_1 + \tfrac{S_3}{S_2} S_0. \end{aligned} \tag{48}$$

(c) If $J = 2$ then:

$$\begin{aligned} S_0 &= Y_1, \\ S_1 &= X_1 Y_1, \\ S_2 &= X_1^2 Y_1 + Y_2, \\ S_3 &= X_1^3 Y_1, \\ S_4 &= X_1^4 Y_1. \end{aligned} \tag{49}$$

Test if the following is true:

$$A_1 \neq A_2 \wedge A_2 \neq A_3 \wedge A_1 = A_4. \tag{50}$$

From (49) follows that:

$$\begin{aligned} X_1 &= \tfrac{S_1}{S_0} = \alpha^j, \\ X_2 &= \alpha^2, \\ Y_1 &= S_0, \\ Y_2 &= S_2 + \tfrac{S_1^2}{S_0}. \end{aligned} \tag{51}$$

(d) If $J = 3$ then:

$$\begin{aligned} S_0 &= Y_1, \\ S_1 &= X_1 Y_1, \\ S_2 &= X_1^2 Y_1, \\ S_3 &= X_1^3 Y_1 + Y_2, \\ S_4 &= X_1^4 Y_1. \end{aligned} \tag{52}$$

Test if the following is true:

$$A_1 = A_2 \wedge A_2 \neq A_3 \wedge A_3 \neq A_4. \tag{53}$$

From (52) follows that:

$$\begin{aligned} X_1 &= \tfrac{S_1}{S_0} = \alpha^j, \\ X_2 &= \alpha^3, \\ Y_1 &= S_0, \\ Y_2 &= S_3 + \tfrac{S_1^3}{S_0^2}. \end{aligned} \tag{54}$$

(e) If $J = 4$ then:

$$\begin{aligned} S_0 &= Y_1, \\ S_1 &= X_1 Y_1, \\ S_2 &= X_1^2 Y_1, \\ S_3 &= X_1^3 Y_1, \\ S_4 &= X_1^4 Y_1 + Y_2. \end{aligned} \tag{55}$$

Test if the following is true:

$$A_1 = A_2 \wedge A_2 \neq A_3 \wedge A_3 \neq A_4. \tag{56}$$

From (55) follows that:

$$\begin{aligned} X_1 &= \tfrac{S_1}{S_0} = \alpha^j, \\ X_2 &= \alpha^4, \\ Y_1 &= S_0, \\ Y_2 &= S_4 + \tfrac{S_1^4}{S_0^3}. \end{aligned} \tag{57}$$

Correction of the received word:

$$\begin{aligned} \hat{c}_j &= v_j + Y_1, \\ \hat{r}_J &= r_J + Y_2, \quad J = 0, 1, \ldots, 4. \end{aligned} \tag{58}$$

(f) In this step an assumption is made that both errors are in the information part. In this case the following is true:

$$A_1 \neq A_2 \wedge A_2 \neq A_3 \wedge A_3 \neq A_4 \tag{59}$$

and the following equations must be fulfilled:

$$S_0 = Y_1 + Y_2, \tag{60}$$

$$S_1 = X_1 Y_1 + X_2 Y_2, \tag{61}$$

$$S_2 = X_1^2 Y_1 + X_2^2 Y_2, \tag{62}$$

$$S_3 = X_1^3 Y_1 + X_2^3 Y_2, \tag{63}$$

$$S_4 = X_1^4 Y_1 + X_2^4 Y_2. \tag{64}$$

The unknown variables X_1, X_2, Y_1, Y_2 can be computed using a standard method described in Section 3. The correction of the received word can be done similarly as in case 4. But now, two positions in the information part have to be corrected corresponding to the two locators X_1 and X_2 by adding the calculated values Y_1 and Y_2 of the errors to the received symbols:

$$\begin{aligned} \hat{c}_i &= v_i + Y_1, \\ \hat{c}_j &= v_j + Y_2. \end{aligned} \tag{65}$$

6. If none of the cases: 1.–6. is confirmed, then it has to be assumed that more than two errors occurred and the decoding failure has to be declared.

In Figure 2 the proposed decoding algorithm is depicted in a compact way using a flow chart.

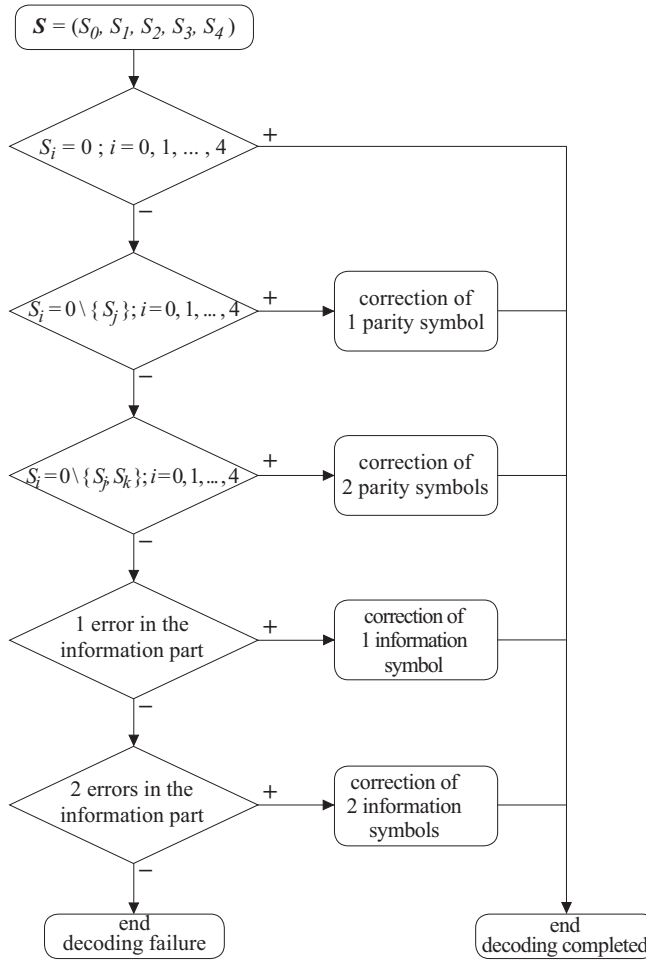

Figure 2. Flow chart of decoding algorithm

5 SOME REMARKS ON DECODING IMPLEMENTATION

In the previous section the basic principles of the decoding algorithm were presented for error correction in five times extended Reed Solomon codes. In this section we will explain some properties which were used in particular steps of the decoding algorithm and describe in more detail how to solve the different, earlier mentioned systems of equations. We will start with polynomial notation for (29) and (32):

$$c(x) = c_{k-1}x^{k-1} + c_{k-2}x^{k-2} + \ldots + c_1x^1 + c_0x^0 + (p_0 + p_1 + p_2 + p_3 + p_4)x^0, \quad (66)$$

$$v(x) = v_{k-1}x^{k-1} + v_{k-2}x^{k-2} + \ldots + v_1x^1 + v_0x^0 + (r_0 + r_1 + r_2 + r_3 + r_4)x^0 \quad (67)$$

which will help to express the formulas needed to explain the proposed decoding algorithm. Similarly the error vector could be expressed using a polynomial:

$$e(x) = e_{k-1}e^{k-1} + e_{k-2}x^{k-2} + \ldots + e_1x^1 + e_0x^0 + (\epsilon_0 + \epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4)x^0. \quad (68)$$

Note: This notation will also be useful later in the section dealing with the analysis of the decoding algorithm complexity. This is because the complexity could be very often decreased by ideas inspired by hardware realizations of decoders. On the other hand the hardware for cyclic codes is conveniently described by polynomials.

By observing (66), (67) and the calculation of the syndromes together with knowing the role of locators during decoding ordinary RS codes, it is obvious that the description (66) and (67) is correct. However, it is not helpful since it does not allow us to distinguish the positions of v_0 and $r_0, r_1, \ldots, r_4$. It is because in this polynomial description all these symbols appear in sum as a coefficient of x^0. Therefore, we will introduce the following new sets of polynomials which are better adapted to the decoding of five times extended RS codes. This approach enables us to solve the mentioned problem with location distinction.

$$\begin{aligned}
c_0(x) &= c_{k-1}x^{k-1} + c_{k-2}x^{k-2} + \ldots + c_1x^1 + c_0x^0 + p_0x^0,\\
c_1(x) &= c_{k-1}x^{k-1} + c_{k-2}x^{k-2} + \ldots + c_1x^1 + c_0x^0 + p_1x^0,\\
c_2(x) &= c_{k-1}x^{k-1} + c_{k-2}x^{k-2} + \ldots + c_1x^1 + c_0x^0 + p_2x^0,\\
c_3(x) &= c_{k-1}x^{k-1} + c_{k-2}x^{k-2} + \ldots + c_1x^1 + c_0x^0 + p_3x^0,\\
c_4(x) &= c_{k-1}x^{k-1} + c_{k-2}x^{k-2} + \ldots + c_1x^1 + c_0x^0 + p_4x^0,
\end{aligned} \quad (69)$$

$$\begin{aligned}
v_0(x) &= v_{k-1}x^{k-1} + v_{k-2}x^{k-2} + \ldots + v_1x^1 + v_0x^0 + r_0x^0,\\
v_1(x) &= v_{k-1}x^{k-1} + v_{k-2}x^{k-2} + \ldots + v_1x^1 + v_0x^0 + r_1x^0,\\
v_2(x) &= v_{k-1}x^{k-1} + v_{k-2}x^{k-2} + \ldots + v_1x^1 + v_0x^0 + r_2x^0,\\
v_3(x) &= v_{k-1}x^{k-1} + v_{k-2}x^{k-2} + \ldots + v_1x^1 + v_0x^0 + r_3x^0,\\
v_4(x) &= v_{k-1}x^{k-1} + v_{k-2}x^{k-2} + \ldots + v_1x^1 + v_0x^0 + r_4x^0,
\end{aligned} \quad (70)$$

$$\begin{aligned}
e_0(x) &= e_{k-1}x^{k-1} + e_{k-2}x^{k-2} + \ldots + e_1x^1 + e_0x^0 + \epsilon_0x^0,\\
e_1(x) &= e_{k-1}x^{k-1} + e_{k-2}x^{k-2} + \ldots + e_1x^1 + e_0x^0 + \epsilon_1x^0,\\
e_2(x) &= e_{k-1}x^{k-1} + e_{k-2}x^{k-2} + \ldots + e_1x^1 + e_0x^0 + \epsilon_2x^0,\\
e_3(x) &= e_{k-1}x^{k-1} + e_{k-2}x^{k-2} + \ldots + e_1x^1 + e_0x^0 + \epsilon_3x^0,\\
e_4(x) &= e_{k-1}x^{k-1} + e_{k-2}x^{k-2} + \ldots + e_1x^1 + e_0x^0 + \epsilon_4x^0.
\end{aligned} \quad (71)$$

Using these polynomials we can express the syndromes as follows:

$$\begin{aligned}
S_0 &= v_0(\alpha^0) = c_0(\alpha^0) + e_0(\alpha^0) = e_{k-1} + \ldots + e_1 + e_0 + \epsilon_0,\\
S_1 &= v_1(\alpha^1) = c_1(\alpha^1) + e_1(\alpha^1) = e_{k-1}\alpha^{k-1} + \ldots + e_1\alpha^1 + e_0\alpha^0 + \epsilon_1\alpha^0,\\
S_2 &= v_2(\alpha^2) = c_2(\alpha^2) + e_2(\alpha^2) = e_{k-1}\alpha^{2(k-1)} + \ldots + e_1\alpha^2 + e_0\alpha^0 + \epsilon_2\alpha^0,\\
S_3 &= v_3(\alpha^3) = c_3(\alpha^3) + e_3(\alpha^3) = e_{k-1}\alpha^{3(k-1)} + \ldots + e_1\alpha^3 + e_0\alpha^0 + \epsilon_3\alpha^0,\\
S_4 &= v_4(\alpha^4) = c_4(\alpha^4) + e_4(\alpha^4) = e_{k-1}\alpha^{4(k-1)} + \ldots + e_1\alpha^4 + e_0\alpha^0 + \epsilon_4\alpha^0.
\end{aligned} \tag{72}$$

Now we can explain some properties on which the particular steps of the proposed decoding algorithm are based depending on error location.

Case 1 is trivial and known for syndrome decoding of linear block codes.

In Case 2, the following property is used: assuming that the error occurs in one of the parity symbols, or in other words, if one of the symbols from a set $\{\epsilon_0, \epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4\}$ in the error polynomial is nonzero, it means that only one corresponding syndrome is nonzero. It follows from (72).

Similarly in Case 3, the following property is used: assuming that only two syndromes e.g. S_I and S_J are nonzero and all others are zeros, then it is supposed that exactly two errors occurred in the parity positions, which follows from (72).

In Case 4, it is supposed that:

$$\epsilon_0 = \epsilon_1 = \epsilon_2 = \epsilon_3 = \epsilon_4 = 0 \tag{73}$$

and only one of the symbols from the set $\{e_{k-1}, e_{k-2}, \ldots, e_1, e_0\}$ is nonzero. From these assumptions it follows that:

$$c_0(x) = c_1(x) = c_2(x) = c_3(x) = c_4(x) = c(x). \tag{74}$$

And therefore (40) follows from (17).

In Case 5, it is supposed that the first error ocured in the information part of the codeword and the second one in the parity part. Therefore, for the first error we can use error locator X_1 as usually defined inside the information part which is mapped on $c(x)$ because in this case: $\epsilon_0 = \epsilon_1 = \epsilon_2 = \epsilon_3 = \epsilon_4 = 0$ and $c_0(x) = c_1(x) = c_2(x) = c_3(x) = c_4(x) = c(x)$. However, for the second error the assumptions are different. Namely that one of the symbols from a set $\{e_{k-1}, e_{k-2}, \ldots, e_1, e_0\}$ in the error polynomial is nonzero. On the other hand, the location and value of the second error is not given by a standard locator in this case. These are determined indirectly using tests: (44),(47),(50),(53),(56) and the system of equations: (45),(48),(51),(54),(57). The reason is that the single nonzero element from the set $\{e_{k-1}, e_{k-2}, \ldots, e_1, e_0\}$ will cause the second summand on the right side to be equal to zero in only one of the above mentioned systems of equations.

In Case 6, the reasoning is similar, because it is assumed that (73) is valid and two of the symbols from set $\{e_{k-1}, e_{k-2}, \ldots, e_1, e_0\}$ are nonzero. Therefore (73) also holds. Consequently (60), (61), (62), (63) and (64) follow from (17) again.

However, in this case it is worth giving more details of the method how to solve the system of Equations (60), (61), (62), (63) and (64). In order to decode the received word it is necessary to find two unknown locators X_1, X_2 and two unknown error values Y_1, Y_2, together 4 unknowns. Therefore at least 4 equations are needed to be calculated. However, the Equations (60), (61), (62), (63) and (64) are not linear. Therefore a direct analytical solution is not easy or viable at all. The following notification allows arguing that the standard approach as in decoding ordinary RS codes is possible. Because in Case 5, it is supposed that two errors occurred, the locator polynomial has degree two:

$$\lambda(x) = \lambda_2 x^2 + \lambda_1 x^1 + x^0. \tag{75}$$

Therefore:

$$\lambda_2 X_i^2 + \lambda_1 X_i^1 + 1 = 0; \quad i = 1, 2. \tag{76}$$

After multiplying (75) with $Y_i X_i^z$; $i = 1, 2$; $z \in Z$ we get:

$$\lambda_2 Y_i X_i^{z+2} + \lambda_1 Y_i X_i^{z+1} + Y_i X_i^z = 0; \quad i = 1, 2 \quad z \in Z. \tag{77}$$

Now we can write (76) for a fixed value of $z \in Z$. For $z = 0$ we get:

$$\begin{aligned} \lambda_2 Y_1 X_1^2 + \lambda_1 Y_1 X_1^1 + Y_1 X_1^0 = 0, \\ \lambda_2 Y_2 X_2^2 + \lambda_1 Y_2 X_2^1 + Y_2 X_2^0 = 0. \end{aligned} \tag{78}$$

Summation of (78) gives us:

$$\lambda_2 S_2 + \lambda_1 S_1 + S_0 = 0. \tag{79}$$

For $z = 1$, we get similarly:

$$\begin{aligned} \lambda_2 Y_1 X_1^3 + \lambda_1 Y_1 X_1^2 + Y_1 X_1^1 = 0, \\ \lambda_2 Y_2 X_2^3 + \lambda_1 Y_2 X_2^2 + Y_2 X_2^1 = 0. \end{aligned} \tag{80}$$

Summation of (80) gives us:

$$\lambda_2 S_3 + \lambda_1 S_2 + S_1 = 0. \tag{81}$$

Now (79) and (81) form a system of linear equations with two unknowns: λ_1 and λ_2. Such a system is easily solvable by standard approaches. For example:

$$\lambda_2 = \frac{S_1 + S_0 S_2}{S_2^2 + S_1 S_3}, \tag{82}$$

$$\lambda_1 = \frac{S_0 + \lambda_2 S_2}{S_1}. \tag{83}$$

As a result we have a concrete error locator polynomial and therefore the Chien algorithm could be used in order to get locators X_1 and X_2. Or (75) can be solved directly [15, 16, 17, 18].

For example in [17] after substituting $x = \frac{\lambda_1}{\lambda_2}u$ into (75), the error locator polynomial will be:

$$\lambda(x) = \frac{\lambda_1^2}{\lambda_2}\left(u^2 + u + \frac{\lambda_2}{\lambda_1^2}\right). \tag{84}$$

In order to find its roots it is necessary to solve:

$$u^2 + u + \frac{\lambda_2}{\lambda_1^2} = 0. \tag{85}$$

Because $GF(2^m)$ is, by squaring, transformed linearly over $GF(2)$, Equation (85) could be reformulated as follows:

$$\mathbf{u}\,(\mathbf{\Theta} + \mathbf{I}) = \frac{\mathbf{\Lambda}_2}{\mathbf{\Lambda}_1^2} \tag{86}$$

where $\mathbf{u}, \frac{\mathbf{\Lambda}_2}{\mathbf{\Lambda}_1^2}, \mathbf{I}$ and $\mathbf{\Theta}$ are binary vectors, identity matrix and $m \times m$ square operator matrix, respectively. After adding the two matrices in (86) the following equality is obtained:

$$\mathbf{u} \times \mathbf{T} = \frac{\mathbf{\Lambda}_2}{\mathbf{\Lambda}_1^2} \tag{87}$$

where $\mathbf{T} = \mathbf{\Theta} + \mathbf{I}$. To make the computation fast the pseudo inverse of $\mathbf{T}$ could be implemented via lookup table in advance for the specific finite field $GF(2^m)$. The calculation of error locators then consists of the following steps:

- calculating $\frac{\mathbf{\Lambda}_2}{\mathbf{\Lambda}_1^2}$;
- reading out the two roots: U_1, U_2 from the lookup table;
- transforming U_1, U_2 into locators X_1, X_2 by using:

$$X_1 = \frac{\mathbf{\Lambda_1}}{\mathbf{\Lambda_2}}U_1; \tag{88}$$

- thanks to Vieta's formulas we can use addition instead of multiplication for the second locator transformation:

$$X_2 = U_1 + \frac{\mathbf{\Lambda_1}}{\mathbf{\Lambda_2}}. \tag{89}$$

After calculating X_1, X_2 they could be inserted into (61). By using the resulting equation and (60) the error values Y_1 and Y_2 could be calculated from this linear system of equations:

$$Y_2 = \frac{S_1 + S_0 X_1}{X_1 + X_2}, \tag{90}$$

$$Y_1 = S_0 + Y_2. \tag{91}$$

Case 7 is a consequence of the fact that all correctable configurations of errors determined by the code distance of the codes are covered by previous cases: 1–6. However, if non correctable configuration of errors occurs, for example containing more than two errors, then the decoding can end in failure or incorrect decoding. The failure can be detected in Case 7. After this detection some remedy mechanism can be started, for example the request for re-sending the codeword again could be generated. Therefore this situation is better than the incorrect decoding caused by more than two errors. This can happen if under the influence of more than two errors the received combination is closer to some other codeword than to one which was sent. Namely its Hamming distance is smaller than or equal to two.

6 COMPLEXITY ESTIMATION OF THE ALGORITHM

In this section the complexity estimation of the proposed decoding algorithm is presented. The number of operations needed for calculating syndromes will not be counted, because these operations are necessary in any syndrome based decoding and they are usually done by hardware. It is obvious that the number of operations necessary for correct decoding is dependent on the number of errors which occur in the transmission channel and on their positions in the received codeword. Therefore an average number of operations per codeword was chosen for this estimation with the assumption that the error occurrence mechanism is modeled as a q-ary symmetric channel depicted in Figure 3.

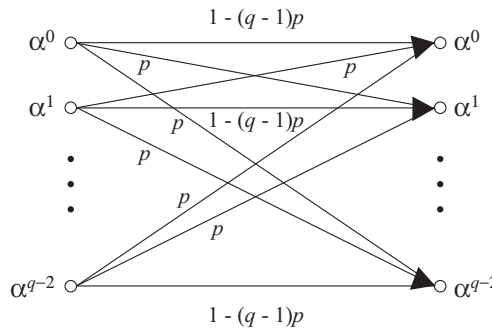

Figure 3. q-ary symmetric channel

If we denote by $P_e = (q-1)p$ the probability of error in this channel model, then:

$$P_0 = (1-P_e)^{q+4}, \tag{92}$$

$$P_1 = \begin{pmatrix} q+4 \\ 1 \end{pmatrix} P_e(1-P_e)^{q+3}, \tag{93}$$

$$P_2 = \begin{pmatrix} q+4 \\ 2 \end{pmatrix} P_e^2(1-P_e)^{q+2} \tag{94}$$

are the probabilities that 0, 1 and 2 errors occur in a received codeword, respectively. However the complexity also depends on the constellation of the error or errors. Therefore the following probabilities for the decoding complexity estimation will be defined:

- probability of one error in parity symbol:

$$P_{1P} = P_1 \frac{5}{q+4} = 5P_e(1-P_e)^{q+3}, \tag{95}$$

- probability of one error in information symbol:

$$P_{1I} = P_1 \frac{q-1}{q+4} = (q-1)P_e(1-P_e)^{q+3}, \tag{96}$$

- probability of two errors in parity symbol:

$$P_{2P} = P_2 \frac{\begin{pmatrix} 5 \\ 2 \end{pmatrix}}{\begin{pmatrix} q+4 \\ 2 \end{pmatrix}} = 10P_e^2(1-P_e)^{q+2}, \tag{97}$$

- probability of two errors in information symbol:

$$P_{2I} = P_2 \frac{\begin{pmatrix} q-1 \\ 2 \end{pmatrix}}{\begin{pmatrix} q+4 \\ 2 \end{pmatrix}} = \frac{(q-1)(q-2)}{2} P_e^2(1-P_e)^{q+2}, \tag{98}$$

- probability of one error in parity symbol and one error in information symbol:

$$P_{1P1I} = P_2 \frac{5(q-1)}{(q+4)^2} = \frac{5(q-1)(q-2)}{2(q+4)} P_e^2(1-P_e)^{q+2}, \tag{99}$$

- probability of more than two errors in codeword:

$$P_{>2} = \sum_{j=3}^{q+4} \left(\begin{array}{c} q+4 \\ j \end{array} \right) P_e^j (1-P_e)^{q+4-j}. \tag{100}$$

After the algorithm gets as an input 5 syndromes, five comparisons of the calculated syndromes are done and depending on the results one of the following subroutines are performed. The complexity for each case can be given by the number of operations. Next, the average number of operations could be calculated using the cumulative probability theorem. We will introduce the vector $\boldsymbol{\Upsilon} = (\upsilon_1, \upsilon_2, \upsilon_3, \upsilon_4, \upsilon_5)$ in which the particular non-negative integer coordinates: $\upsilon_1, \upsilon_2, \upsilon_3, \upsilon_4, \upsilon_5$ denote the number of additions, multiplications, divisions, comparisons and look up table accesses, respectively, which are needed in order to perform the particular subroutines. The detailed estimation of the necessary operations (except calculation of syndromes) follows:

- one error in parity symbol: 1 addition:

$$\boldsymbol{\Upsilon}_{1P} = (1, 0, 0, 0, 0), \tag{101}$$

- two errors in parity symbol: 2 additions:

$$\boldsymbol{\Upsilon}_{2P} = (2, 0, 0, 0, 0), \tag{102}$$

- one error in information symbol: 1 addition, 4 divisions and 3 comparisons:

$$\boldsymbol{\Upsilon}_{1I} = (1, 0, 4, 3, 0), \tag{103}$$

- one error in parity symbol and one error in information symbol. Now the calculations and mutual comparison of auxiliary variables: A_1, A_2, A_3, A_4 was done in the previous case. In the worst case 3. additional additions, one additional multiplication and 3 additional divisions have to be performed. Therefore:

$$\boldsymbol{\Upsilon}_{1P1I} = (4, 1, 7, 3, 0), \tag{104}$$

- two errors in information symbols: For the calculations and mutual comparison of auxiliary variables: A_1, A_2, A_3, A_4 holds the same as in the previous case. Then the correction of two errors will need for solutions of (88), (89),(90) and (91): 4 additions, 8 divisions, 5 multiplications and 2 readouts from the look up table:

$$\boldsymbol{\Upsilon}_{2I} = (4, 5, 12, 3, 2), \tag{105}$$

- more than two errors in information symbol: In this case the complexity of procedures quantified by $\boldsymbol{\Upsilon}_{1I}, \boldsymbol{\Upsilon}_{2I}, \boldsymbol{\Upsilon}_{1P1I}$ can be added as the worst case:

$$\boldsymbol{\Upsilon}_{>2I} = (16, 6, 23, 9, 2). \tag{106}$$

We will characterize the overall complexity using a vector: $\mathbf{\Gamma} = (\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5)$ in which the particular coordinates: $\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5$ denote average numbers of: additions, multiplications, divisions, comparisons and look up table accesses, respectively, which are needed in order to perform the particular subroutines. The value of $\mathbf{\Gamma}$ can be evaluated from (101), (102), (103), (104), (105) and (106) using the cumulative probability formula:

$$\begin{aligned}\mathbf{\Gamma} = &\ \mathbf{\Upsilon}_{1P} \times 5P_e(1-P_e)^{q+3} \\ &+ \mathbf{\Upsilon}_{2P} \times 10P_e^2(1-P_e)^{q+2} \\ &+ \mathbf{\Upsilon}_{1I} \times (q-1)P_e(1-P_e)^{q+3} \\ &+ \mathbf{\Upsilon}_{2I} \times \frac{(q-1)(q-2)}{2}P_e^2(1-P_e)^{q+2} \\ &+ \mathbf{\Upsilon}_{1P1I} \times \frac{5(q-1)(q-2)}{2(q+4)}P_e^2(1-P_e)^{q+2} \\ &+ \mathbf{\Upsilon}_{>2I} \times \sum_{j=3}^{q+4} \left(\begin{array}{c} q+4 \\ j \end{array} \right) P_e^j(1-P_e)^{q+4-j}. \end{aligned} \tag{107}$$

After evaluation using (107) it is possible to get information about the average number of particular operations: additions, multiplications, divisions, comparisons and accessing the look up table, necessary for decoding the received codewords. By observing (107) it is also obvious that these average numbers of operations depend on the error probability P_e of the q-ary symmetric channel and on q – the number of elements in the finite field.

The total average number of operations per codeword is given by the sum of all coordinates of (107):

$$\bar{\mathbf{\Gamma}} = \sum_{i=1}^{5} \gamma_i. \tag{108}$$

A tabular representation of (108) for selected values of P_e for $GF(64)$ and $GF(256)$ is shown in Table 1.

	GF(64)	GF(256)
P_e	Γ	Γ
10^{-2}	7.058	34.89
10^{-3}	5.280×10^{-1}	2.370
10^{-4}	5.109×10^{-2}	2.077×10^{-1}
10^{-5}	5.092×10^{-3}	2.048×10^{-2}

Table 1. The average number of operations

It has to be stressed out that the complexity estimations in Table 1 do not include the complexities of syndrome calculations. These complexities are constant

for particular finite fields. In $GF(q)$ if we use the Horner scheme for syndrome calculations we will need $5 \times (q-2)$ additions and $5 \times (q-2)$ multiplications. E.g. in $GF(64)$ and in $GF(256)$ the overall number of operations needed for calculation of syndromes will be 640 and 2 560, respectively. As was already mentioned, the syndromes always have to be calculated in any syndrome decoding algorithm and usually this is done via hardware. The average number of operations needed in the proposed algorithm for example where $P_e = 10^{-3}$ is less than 1 and less than 3 in $GF(64)$ and $GF(256)$, respectively.

7 CONCLUSIONS

In this paper a new decoding algorithm was proposed for the five times extended Reed Solomon codes proposed in [1], which allows correcting up to two errors in a codeword. In contrast to known error correcting syndrome decoding algorithms, the method proposed in this paper has to deal with the problem that the control matrix is not a pure Vandermonde matrix, but it is a juxtaposition of a Vandermonde and an identity matrix. Therefore, the proposed algorithm had to (to some extent) use strategy somewhat like a set of sieves in order to separate different possible error patterns. This strategy is explained in detail using specially created polynomial and matrix analytical descriptions. This algorithm was also verified using Mathematica software for all distinct cases and all typical combinations of error patterns.

Acknowldegement

This work was supported by the Slovak Research and Development Agency under the Contract No. APVV-19-0436 and it was also supported by the Scientific Grant Agency of the Ministry of Education of the Slovak Republic and the Slovak Academy of Sciences (grant VEGA No. 1/0477/18).

REFERENCES

[1] Rakús, M.—Farkaš, P.—Páleník, T.—Daniš, A.: Five Times Extended Reed-Solomon Codes Applicable in Memory Storage Systems. IEEE Letters of the Computer Society, Vol. 2, 2019, No. 2, pp. 9–11, doi: 10.1109/LOCS.2019.2911517.

[2] Rakús, M.—Farkaš, P.—Páleník, T.: Erasure Decoding of Five Times Extended Reed-Solomon Codes. Journal of Electrical Engineering, Vol. 70, 2019, No. 3, pp. 256–258, doi: 10.2478/jee-2019-0035.

[3] Reed, I. S.—Solomon, G.: Polynomial Codes over Certain Finite Fields. M.I.T. Lincoln Laboratory Group Report 47.23, 31 December 1958.

[4] Wicker, S. B.—Bhargava, V. K.: Reed-Solomon Codes and Their Applications. Wiley-IEEE Press, 1999, 336 pp., doi: 10.1109/9780470546345. ISBN 978-0-780-35391-6.

[5] Li, W.—Wang, Z.—Jafarkhani, H.: Repairing Reed-Solomon Codes over $GF(2^l)$. IEEE Communications Letters, Vol. 24, 2020, No. 1, pp. 34–37, doi: 10.1109/LCOMM.2019.2950922.

[6] Zhang, G.—Liu, H.: Constructions of Optimal Codes with Hierarchical Locality. IEEE Transactions on Information Theory (Early access), doi: 10.1109/TIT.2020.2977636.

[7] Chen, Z.—Barg, A.: Explicit Constructions of MSR Codes for Clustered Distributed Storage: The Rack-Aware Storage Model. IEEE Transactions on Information Theory, Vol. 66, 2020, No. 2, pp. 886–899, doi: 10.1109/TIT.2019.2941744.

[8] Zhang, Y.—Zhang, Z.: An Improved Cooperative Repair Scheme for Reed-Solomon Codes. Proceedings of 2019 19th International Symposium on Communications and Information Technologies (ISCIT), Ho Chi Minh City, Vietnam, 2019, pp. 525–530, doi: 10.1109/ISCIT.2019.8905159.

[9] Li, W.—Wang, Z.—Jafarkhani, H.: On the Sub-Packetization Size and the Repair Bandwidth of Reed-Solomon Codes. IEEE Transactions on Information Theory, Vol. 65, 2019, No. 9, pp. 5484–5502, doi: 10.1109/TIT.2019.2917425.

[10] Jagmohan, A.—Lastras-Montano, L. A. (Inventors): Mis-Correction and No-Correction Rates for Error Control. US patent: US 8 806 295 B2, Aug. 12, 2014 (Assignee: IBM Corp.).

[11] Rakús, M.—Farkaš, P.: Double Error Correcting Codes with Improved Code Rates. Journal of Electrical Engineering, Vol. 55, 2004, No. 3-4, pp. 89–94.

[12] Berlekamp, E. R.: Nonbinary BCH Decoding. International Symposium on Information Theory, San Remo, Italy, 1967, doi: 10.1109/tit.1968.1054109.

[13] Massey, J.: Shift-Register Synthesis and BCH Decoding. IEEE Transactions on Information Theory, Vol. 15, 1969, No. 1, pp. 122–127, doi: 10.1109/TIT.1969.1054260.

[14] Chien, R.: Cyclic Decoding Procedures for Bose-Chaudhuri-Hocquenghem Codes. IEEE Transactions on Information Theory, Vol. 10, 1964, No. 4, pp. 357–363, doi: 10.1109/TIT.1964.1053699.

[15] Berlekamp, E. R.—Rumsey, H.—Solomon, G.: On the Solution of Algebraic Equations over Finite Fields. Information and Control, 1967, pp. 553–564, doi: 10.1016/s0019-9958(67)91016-9.

[16] Walker, C. W.: New Formulas for Solving Quadratic Equations over Certain Finite Fields. IEEE Transactions on Information Theory, Vol. 45, 1999, No. 1, pp. 283–284, doi: 10.1109/18.746816.

[17] Pommerening, K.: Quadratic Equations in Finite Fields of Characteristic 2. May 2000, English Version, February 2012, available online: `https://www.staff.uni-mainz.de/pommeren/MathMisc/QuGlChar2.pdf`.

[18] Goresky, M.—Klapper, A.: Algebraic Shift Register Sequences. Cambridge University Press, 2005.

Peter Farkaš is with the Institute of Multimedia Information and Communication Technologies, Slovak University of Technology in Bratislava (STU) and also with the Institute of Applied Informatics, Faculty of Informatics, Pan European University in Bratislava as Professor. From 2002 until 2007 he was Visiting Professor at Kingston University, UK and Senior Researcher at SIEMENS PSE. In 2003 SIEMENS named him VIP for his innovations and patents. In 2004 he was awarded with the Werner von Siemens Excellence Award for research results on two-dimensional complete complementary codes. From 2008 to 2009 he worked also as Consultant in the area of software defined radio for SANDBRIDGE Tech. (USA). He was the Leader of a Team from STU in projects funded by the European Community under the 5FP and 6FP "Information Society Technologies Programs": NEXWAY IST-2001-37944 (Network of Excellence in Wireless Applications and Technology) and CRUISE (Creating Ubiquitous Intelligent Sensing Environments) FP6 IST-2005-4-027738 (2006–2007). His research interests include coding, communications theory and sequences for CDMA. He has published 1 book, about 45 papers in reviewed scientific journals and about 100 papers in international conferences. He is the author or co-author of 7 patents. He is and was serving in TPC of about 60 international conferences and presented 12 invited lectures. As an IEEE volunteer, he was serving in the IEEE Czechoslovakia Section Executive Committee in different positions from 1992 to 2014 and from 2005 to 2006 he served as Chair of the Conference Coordinator Subcommittee in IEEE Region 8. He organized the IEEE R8 Conference EUROCON 2001 and was Chairman of SympoTIC '03, SympoTIC '04, SympoTIC '06 and co-organizer of the Winter School on Coding and Information Theory 2005. From 2016 he has been serving as a vice-chair of the Computer chapter in the IEEE Czechoslovakia Section.

Martin Rakús studied radio electronics and graduated from the Slovak University of Technology in 2001. In 2004 he received his Ph.D. from the Slovak University of Technology and in 2020 he became Associate Professor at the same institute. Since 1995 he has been with the Institute of Multimedia Information and Communication Technology, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovakia. His primary research interests are error control coding and digital communication systems. He is a member of the IEEE.