REED-SOLOMON CODE SYMBOL AVOIDANCE

T. Shongwe* **and A. J. Han Vinck**†

* *Department of Electrical and Electronic Engineering Science, University of Johannesburg, P.O. Box 524, Auckland Park, 2006, Johannesburg, South Africa E-mail: tshongwe@uj.ac.za*

† *University of Duisburg-Essen, Institute for Experimental Mathematics, Ellernstr. 29, 45326 Essen, Germany E-mail: vinck@iem.uni-due.de*

Abstract: A Reed-Solomon code construction that avoids or excludes particular symbols in a linear Reed-Solomon code is presented. The resulting code, from our *symbol avoidance* construction, has the same or better error-correcting capabilities compared to the original Reed-Solomon code, but with reduced efficiency in terms of rate. The codebook of the new code is a subset of the original Reed-Solomon code and the code may no longer be linear. We also present computer search results for the bound on the number of symbols that can be avoided, and we make an attempt to find an expression for the bound. Such a code, by symbol avoidance, can be well suited to a number of applications, some of which include markers for synchronization, frequency hopping signatures, and pulse position modulation.

Key words: Reed-Solomon codes, Subcodes.

1. INTRODUCTION

In the work presented by Solomon [1] on alphabet codes and fields of computation, he showed the following. Given an alphabet (Q) and a field of computation (F) which is prime or power of a prime, where the size of Q is less than that of F, it is possible to form a code over Q from another code over F with the field of computation of the encoding procedure being F. The field F is defined as the field of encoding because that is where the encoding process takes place and the resulting code being over Q. It was also shown in [1] that the price to pay for this modification in the code is reduction in efficiency in the code over Q, when comparing it to the corresponding code over F. A fitting example of a code over the field F was taken as an (n, k, d) Reed-Solomon (RS) code, which has well defined encoding and decoding operations, and is known to have good error-correcting capabilities. In the (n, k, d) RS code, n is the length, k is the dimension and d is the minimum Hamming distance of the code. The reader who wants to get acquainted with the principles of Reed-Solomon encoding is referred to [2] and [3]. Solomon [1] used the example of the RS code over F (F is normally a Galois field–GF) to show that from the RS code, a code over alphabet Q can be derived which is "almost" a RS code.

The motivation for the work by Solomon [1] was the fact that not all alphabets we encounter in systems are of the same size as the field F in which most codes are defined over. For example, the decimal number system with the alphabet $\{0,1,\ldots,9\}$, can have GF(11) as the field of computation for the encoding, and the English alphabet with 26 letters, can have GF(3^3) or GF(29) as the field of computation for the encoding. This idea can therefore be seen as reducing the size of the field F to form codes over a smaller alphabet Q, or avoiding using some of the symbols of the field F in the alphabet Q. Solomon [4] did another work, related to his previous work in [1], where he introduced non-linear, non-binary cyclic group codes. These codes were associated with GF(2^m) such that they were of length $2^m - 1$, but had $(m - j)$-bit symbols instead of the usual m-bit symbols, for m and j positive integers. Further work, in which Solomon was one of the authors, was done on subspace subcodes of Reed-Solomon codes in [5]. The work in [5] presented RS codes that were subsets of parent RS codes, and the main contribution was coming up with a formula for the dimension for such codes. The work in [5] was limited to RS codes over GF(2^m). Recently, Urivskiy [6] published work on subset subcodes of linear codes. The subcodes mentioned in [6] were over a set which was a subset of GF(q), where q is power of a prime. This idea was similar to the idea in [1]. However, the focus of the work in [6] was not specifically on RS codes, it was on linear block codes and finding bounds on the cardinality of the subcodes.

In this paper we take the idea in [1] further; we present an encoding procedure by which new codes can be formed over an alphabet that is smaller than the size of the field of computation. We call this procedure *symbol avoidance* because it allows for the avoidance of particular symbols in the new code. Our symbol avoidance procedure is a more generalised case of the construction in [1]. We will also attempt to give an expression for a bound on the number of symbols that can be avoided for each code and show why it may not be possible to get a good estimate of the bound. The codes resulting from our symbol avoidance procedure of a RS code may have the same distance properties as the original RS code or better and may be non-linear.

The applications to symbol avoidance include those mentioned in [1] and the following. Frame synchronization: this involves using the avoided symbols for synchronization to mark the start/end of a frame, or as pilot symbols. Frequency hopping signatures and pulse position modulation: in multiple access communications where each user is given a unique signature to communicate, it may be desired to avoid particular symbols from occurring in the signatures in order to avoid or minimise conflicts/collision among users.

2. SYMBOL AVOIDANCE

Let us define a linear RS code as $(n,\ k,\ d)W$ over GF(q), where n is the length, k is the dimension, d is the minimum Hamming distance and q is the size of the field which is power of a prime. From the linear RS code $(n,\ k,\ d)W$ we produce a new code $(n,\ k',\ d')W'$, of length n, dimension k' and minimum Hamming distance d', over an alphabet of size q'. We call the operation by which W' is produced from W, symbol avoidance. This operation is given in simplified form as

$$(n,\ k,\ d)W \rightarrow \left\{ \begin{matrix} \textit{Symbol} \\ \textit{Avoidance} \\ \textit{Operation} \end{matrix} \right\} \rightarrow (n,\ k',\ d')W',$$

where $d' \geq d$, $q' < q$, $k' < k$, and $(n,\ k',\ d')W'$ may be non-linear. $q' = q - |A|$, where A is a set of elements/symbols to be avoided in $(n,\ k',\ d')W'$. Next, we explain the symbol avoidance operation in detail.

The conventional systematic generator matrix of the RS code $(n,\ k\ d)W$, $G = [I_k|P_{n-k}]$ with the symbols taken from a Galois field GF(q), is decomposed into two parts. The first part of G, which is composed of k' rows of G, will be denoted $G^{k'}$. The second part of G, which is composed of r rows of G such that $k = k' + r$, will be denoted G^r.

$$G = \begin{bmatrix} I_{k'} & 0^{k'}_{r} & | \, P^{k'}_{n-k} \\ 0^{r}_{k'} & I_r & | \, P^{r}_{n-k} \end{bmatrix},$$

where $G^{k'} = [I_{k'}\ 0^{k'}_{r}|P^{k'}_{n-k}]$, $G^r = [0^r_{k'}\ I_r|P^r_{n-k}]$ and

$$P^{k'}_{n-k} = \begin{bmatrix} P^{k'}_{11} & P^{k'}_{21} & \cdots & P^{k'}_{1(n-k)} \\ \vdots & \vdots & & \vdots \\ P^{k'}_{k'1} & P^{k'}_{k'2} & \cdots & P^{k'}_{k'(n-k)} \end{bmatrix}, \tag{1}$$

$$P^{r}_{n-k} = \begin{bmatrix} P^{r}_{11} & P^{r}_{21} & \cdots & P^{r}_{1(n-k)} \\ \vdots & \vdots & & \vdots \\ P^{r}_{r1} & P^{r}_{r2} & \cdots & P^{r}_{r(n-k)} \end{bmatrix}. \tag{2}$$

$G^{k'}$ is used to encode a k'-tuple ($M = m_1 m_2 \ldots m_{k'}$, where $m_i \in$ GF(q)), and this results into a codeword $C = MG^{k'}$. G^r encodes an r-tuple ($V = v_1 v_2 \ldots v_r$, where $v_i \in$ GF(q)), resulting into what we shall call a *control vector* $R = VG^r$. The difference between C and R will be in their usage, otherwise they are both results of RS encoding. The control vectors (collection of the vectors R) are used to control the presence/absence of a particular symbol(s) in each codeword C, as will be demonstrated shortly. For a q-ary linear block code with a systematic generator matrix, *undesired symbols* in the codeword, due to the identity part, can be avoided by simply not including those symbols in the message to be encoded. However, for the parity part of the generator, a different method to avoid undesired symbols needs to be applied. It is therefore the main task of this paper to show that we can avoid undesired symbols in a Reed-Solomon code while maintaining or improving its minimum Hamming distance even though the new code may be non-linear. Using control vectors, we control which symbols to avoid in the parity part of the RS code. C is the RS codeword and R is a control vector to be used on C, in case C has an undesired symbol.

We now focus on the parity parts of C and R. We denote by $P^C_1, P^C_2, \ldots P^C_{n-k}$ and $P^R_1, P^R_2, \ldots P^R_{n-k}$ the parity symbols for C and R, respectively. Using (1), each parity symbol of a codeword, P^C_i, becomes $P^C_i = m_1 P^{k'}_{1i} + \ldots + m_{k'} P^{k'}_{k'i}$, for $1 \leq i \leq n-k$. Using (2), each parity symbol of a control vector, P^R_j, becomes $P^R_j = v_1 P^r_{1j} + \ldots + v_r P^r_{rj}$, for $1 \leq j \leq n-k$. Let us define a set $A = \{a_1, a_2, \ldots, a_{|A|}\}$, which is a set of symbols taken from GF(q), where $|.|$ here indicates the cardinality of a set. The symbols in set A are the symbols we want to avoid in all codewords that resulted from the encoding by $G^{k'}$. If any $P^C_i \in A$ then we ought to eliminate/avoid that P^C_i using a corresponding P^R_j, where $j = i$. This is done by adding the corresponding parity parts of C and R such that the resulting parity symbols in the new codeword, P_i $(= P^C_i + P^R_i)$, do not have any of the symbols in A. This procedure can be expressed mathematically as $P_i \neq a_x$, or $P^C_i + P^R_i \neq a_x$, for $1 \leq x \leq |A|$ and $1 \leq i \leq n-k$. Depending on their suitability in a sentence, the words eliminate and avoid will be used interchangeably since they convey the same message.

The next example illustrates the symbol avoidance procedure.

Example 1 *For this example, we take an $(n = 7,\ k = 3)$ RS code over GF(2^3) with the generator G in* (3). *The field GF(2^3) is generated by a primitive polynomial, $p(I) = I^3 + I + 1$.*

$$G = \begin{bmatrix} 1 & 0 & 0 & | & 6 & 1 & 6 & 7 \\ 0 & 1 & 0 & | & 4 & 1 & 5 & 5 \\ 0 & 0 & 1 & | & 3 & 1 & 2 & 3 \end{bmatrix}. \tag{3}$$

Then by choosing $r = 1$, we have $k' = 2$,

$$G^{k'} = \begin{bmatrix} 1 & 0 & 0 & | & 6 & 1 & 6 & 7 \\ 0 & 1 & 0 & | & 4 & 1 & 5 & 5 \end{bmatrix} \tag{4}$$

and

$$G^r = \begin{bmatrix} 0 & 0 & 1 & | & 3 & 1 & 2 & 3 \end{bmatrix}. \tag{5}$$

Let 7 *be the symbol to avoid in codewords encoded by $G^{k'}$ in* (4)*, hence $A = \{7\}$.*

The G^r in (5) *gives the following control vectors we can use when we encounter a codeword with symbol* 7*,*

$$\begin{bmatrix} 0 & 0 & 0 & \mid & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \mid & 3 & 1 & 2 & 3 \\ 0 & 0 & 2 & \mid & 6 & 2 & 4 & 6 \\ 0 & 0 & 3 & \mid & 5 & 3 & 6 & 5 \\ 0 & 0 & 4 & \mid & 7 & 4 & 3 & 7 \\ 0 & 0 & 5 & \mid & 4 & 5 & 1 & 4 \\ 0 & 0 & 6 & \mid & 1 & 6 & 7 & 1 \\ 0 & 0 & 7 & \mid & 2 & 7 & 5 & 2 \end{bmatrix}. \qquad (6)$$

The last control vector, $[0\ 0\ 7 \mid 2\ 7\ 5\ 2]$*, is included for completeness, otherwise it cannot be used to eliminate symbol* 7 *since it has* 7 *in its information part. To illustrate the elimination of symbol* 7*, we pick one codeword with the symbol* 7 *(in the parity part) from the list of codewords produced by $G^{k'}$, and that codeword is* $C = [0\ 3\ 0 \mid 7\ 3\ 4\ 4]$*. From C, we have* $P^C_1 = 7$*,* $P^C_2 = 3$*,* $P^C_3 = 4$ *and* $P^C_4 = 4$*. Taking the control vector to use as* $R = [0\ 0\ 1 \mid 3\ 1\ 2\ 3]$ *from* (6)*, we have* $P^R_1 = 3$*,* $P^R_2 = 1$*,* $P^R_3 = 2$ *and* $P^R_4 = 3$*. Adding the corresponding parity symbols, we have* $P_1 = P^C_1 + P^R_1 = 4$*,* $P_2 = P^C_2 + P^R_2 = 2$*,* $P_3 = P^C_3 + P^R_3 = 6$ *and* $P_4 = P^C_4 + P^R_4 = 7$*. The new codeword has* $P_1 = 4$*,* $P_2 = 2$*,* $P_3 = 6$ *and* $P_4 = 7$*, which still contains symbol* 7*, hence* $R = [0\ 0\ 1 \mid 3\ 1\ 2\ 3]$ *is not a suitable control vector. We repeat this procedure with every control vector in* (6) *until we find one that is suitable, and in this case it is* $R = [0\ 0\ 3 \mid 5\ 3\ 6\ 5]$*, which gives* $P_1 = 2$*,* $P_2 = 0$*,* $P_3 = 2$ *and* $P_4 = 1$*. It can also be verified that* $R = [0\ 0\ 5 \mid 4\ 5\ 1\ 4]$*,* $[0\ 0\ 6 \mid 1\ 6\ 7\ 1]$ *and* $[0\ 0\ 2 \mid 6\ 2\ 4\ 6]$ *are all suitable control vector.*

Table 1: Different representations of the elements of GF(2^3) generated by the primitive polynomial $p(I) = I^3 + I + 1$, where α is a primitive element of GF(2^3).

Elements of GF(2^3)	Binary representation	Polynomial representation	Decimal representation of the binary
0	000		0
1	001	1	1
α	010	α	2
α^2	100	α^2	4
α^3	011	$\alpha+1$	3
α^4	110	$\alpha^2+\alpha$	6
α^5	111	$\alpha^2+\alpha+1$	7
α^6	101	α^2+1	5

Remark: *It is common practice to represent the elements of GF(q) by the powers of some primitive element, say* α*. For ease of presentation, we represented the elements of GF(q) in decimal format in this example. Throughout this paper, we use the decimal representation to represent the elements of GF(q). The operations are still done the conventional way using the primitive element. As an example, the relationship between the powers of* α *and their decimal equivalents is shown in Table 1, where the elements of GF(2^3) are generated by a primitive polynomial* $p(I) = I^3 + I + 1$. ■

Note that the key operation of the symbol avoidance procedure, we just demonstrated with in Example 1, is about adding two codewords of the original RS code W to form a new codeword without the symbols in A. This new codeword then belongs to W' together with all other codewords without the symbols in A. Since W is a linear code, then it always holds that $W' \subseteq W$. According to the definition of linear codes, W' can be non-linear, but its minimum Hamming distance can be the same as that of W or even greater.

2.1 Table Format Representation

Having shown in Example 1 that some control vectors are not suitable for symbol avoidance for a particular codeword, we now want to look at the number of such control vectors. To do this we take a look at the r-tuples that generate these unsuitable control vectors. The list of all possible combinations of $v_1v_2\ldots v_r$ (r-tuples) generating the unsuitable control vectors given a_x and P_i^C, is tabulated. Remember that the situation $P_i^C + P_i^R = a_x$ or $P_i^R = a_x - P_i^C$ should be avoided, which is a representation of a list of P_i^R that should be avoided. Again, remembering that $P_j^R = v_1P^r_{1j} + \ldots + v_rP^r_{rj}$, then $P_i^R = v_1P^r_{1i} + \ldots + v_rP^r_{ri} = a_x - P_i^C$. Since a_x and P_i^C are given, the task is to find a list of all possible r-tuples ($v_1v_2\ldots v_r$) for which $P_i^R = a_x - P_i^C$ is satisfied. This will give us all the r-tuples to be avoided for each codeword, given set A. Each r-tuple, to be avoided, multiplied by G^r corresponds to a control vector that is not suitable for symbol avoidance.

At this point we want to stress the following. All the possible r-tuples when multiplied by G^r, give a list of available control vectors. Among the available control vectors, there are those that are suitable and those that are unsuitable for symbol avoidance. This means that some r-tuples generate control vectors not suitable for symbol avoidance (r-tuples satisfying $P_i^R = a_x - P_i^C$), and hence have to be avoided. The remainder of the r-tuples generate suitable control vectors (r-tuples satisfying $P_i^R \neq a_x - P_i^C$). We will pay more attention to the r-tuples satisfying $P_i^R = a_x - P_i^C$ (r-tuples to be avoided), as they indicate to us when symbol avoidance is not possible. It will be more clear later why these r-tuples satisfying $P_i^R = a_x - P_i^C$ are important.

Using Example 1, a list of all the possible combinations of $v_1v_2\ldots v_r$ satisfying each $a_x - P_i^C$ is found and tabulated as shown in Table 2. The field of computation is GF(2^3) as before, and since the field is an extension of a binary field where addition and subtraction mean the same thing, then $a_x - P_i^C$ is the same as $a_x + P_i^C$. Since $r = 1$ and $|A| = 1$,

$$v_1P^r_{1i} = a_1 + P_i^C. \qquad (7)$$

Then for $C = [0\ 3\ 0 \mid 7\ 3\ 4\ 4]$ and $a_1 = 7$, for each of the ($n-k=4$) parity symbols, the equations to solve according to (7) are listed as follows.

$$\begin{aligned} 3v_1 &= 7+7 \\ v_1 &= 0. \end{aligned} \tag{8}$$

$$\begin{aligned} v_1 &= 7+3 \\ v_1 &= 4. \end{aligned} \tag{9}$$

$$\begin{aligned} 2v_1 &= 7+4 \\ v_1 &= 4. \end{aligned} \tag{10}$$

$$\begin{aligned} 3v_1 &= 7+4 \\ v_1 &= 1. \end{aligned} \tag{11}$$

The solution of each equation gives the r-tuple (one symbol in this case) to be avoided for each corresponding parity symbol in the codeword, given set A. In this particular example, we have only $v_1 = 0$ or 4 or 1 to be avoided, which means only three control vectors cannot be used. We will refer to the r-tuple(s) to be avoided for each codeword, given set A, simply as *r-tuple to be avoided*. The results of equations (8)–(11) are summarised in Table 2.

Table 2: Table listing all r-tuples to be avoided.

$P_1^C = 7$	$P_2^C = 3$	$P_3^C = 4$	$P_4^C = 4$
$a_1 = 7$	$a_1 = 7$	$a_1 = 7$	$a_1 = 7$
v_1	v_1	v_1	v_1
0	4	4	1

We shall use this table format (Table 2) as it makes analysis of results, for larger numbers of r-tuples to be avoided, easier. The next example will demonstrate this. Using this table format will help in an attempt to derive an intuitive bound on $|A|$, as will be shown later on.

Example 2 *Using the same RS code in Example 1, but now having $r = 2$, $A = \{5,6,7\}$, $|A| = 3$ and codeword, $C = [0\,0\,4 \mid 7\,4\,3\,7]$. With $r = 2$, we have $k' = 1$,*

$$G^{k'} = \begin{bmatrix} 0 & 0 & 1 & \mid & 3 & 1 & 2 & 3 \end{bmatrix}$$

and

$$G^{r} = \begin{bmatrix} 1 & 0 & 0 & \mid & 6 & 1 & 6 & 7 \\ 0 & 1 & 0 & \mid & 4 & 1 & 5 & 5 \end{bmatrix}.$$

Note: *Here we intentionally set the last row of G in* (3) *to $G^{k'}$ and the remaining rows of G to G^r, to show that it does not matter which rows of G are assigned to $G^{k'}$ and G^r.*

For each P_i^C and each a_x, the equation for all possible combinations of v_1 and v_2 to be avoided is given by $v_1P_{1i}^r + v_2P_{2i}^r = a_x + P_i^C$. The results of all r-tuples to be avoided are given in Table 3.

Table 3: Table listing all r-tuples to be avoided, for $n-k = 4$, $r = 2$ and $A = \{5,6,7\}$.

	$P_1^C = 7$			$P_2^C = 4$			$P_3^C = 3$			$P_4^C = 7$		
	$a_1 = 5$	$a_1 = 6$	$a_1 = 7$	$a_1 = 5$	$a_1 = 6$	$a_1 = 7$	$a_1 = 5$	$a_1 = 6$	$a_1 = 7$	$a_1 = 5$	$a_1 = 6$	$a_1 = 7$
	v_1 v_2	v_1 v_2	v_1 v_2	v_1 v_2	v_1 v_2	v_1 v_2	v_1 v_2	v_1 v_2	v_1 v_2	v_1 v_2	v_1 v_2	v_1 v_2
$q^{r-1} = 8$	6 0	3 0	0 0	1 0	2 0	3 0	1 0	4 0	7 0	3 0	4 0	0 0
	1 1	4 1	7 1	0 1	3 1	2 1	5 1	0 1	3 1	1 1	6 1	2 1
	3 2	6 2	5 2	3 2	0 2	1 2	2 2	7 2	4 2	7 2	0 2	4 2
	4 3	1 3	2 3	2 3	1 3	0 3	6 3	3 3	0 3	5 3	2 3	6 3
	7 4	2 4	1 4	5 4	6 4	7 4	7 4	2 4	1 4	0 4	7 4	3 4
	0 5	5 5	6 5	4 5	7 5	6 5	3 5	6 5	5 5	2 5	5 5	1 5
	2 6	7 6	4 6	7 6	4 6	5 6	4 6	1 6	2 6	4 6	3 6	7 6
	5 7	0 7	3 7	6 7	5 7	4 7	0 7	5 7	6 7	6 7	1 7	5 7

$|A|(n-k) = 3\times4 = 12$

It can be seen in Table 3 that the total number of the r-tuples to be avoided is the product of the following:

- *the number of symbols in set A: $|A|$,*
- *the number of parity symbols in the RS code: $n-k$,*
- *and the number of unique r-tuples with symbols from GF(q): q^{r-1}.*

This results in the expression, $|A|(n-k)q^{r-1}$. ■

2.2 Bound

The previous subsection addressed the matter of counting the r-tuples to be avoided, given a symbol(s) to be avoided (set A) and a codeword of the RS code. An interesting question to be answered is, what is the limit to the number of symbols that can be avoided given a RS code when given r? An answer or an attempt to provide an answer to this question involves estimating an upper bound on $|A|$. We have shown that the number of r-tuples (or number of control vectors) to be avoided given $a_x \in A$, is $|A|(n-k)q^{r-1}$. We also know that the control vectors generated by the combinations of $v_1v_2\ldots v_r$ when given q is q^r. However, with the condition that $|A|$ symbols are to be avoided, only $(q-|A|)$ symbols will be available to generate control vectors from sequences of length r. The number of available control vectors is then, $(q-|A|)^r$, and the number of control vectors that are not suitable for symbol avoidance (generated by the r-tuples to be avoided) is $|A|(n-k)q^{r-1}$. It stands to reason that for successful avoidance of symbols, the control vectors not suitable for symbol avoidance should be less than the total number of available control vectors. This leads to

$$|A|(n-k)q^{r-1} < (q-|A|)^r, \tag{12}$$

as a sufficient condition for symbol avoidance.

We know that given a linear RS code $(n,\ k\ d)W$ and r, we can apply the symbol avoidance operation on W such that the resulting code is $(n,\ k'\ d')W'$, where $k' = k-r$ and $d' \geq d$. We now attempt to estimate an upper bound on $|A|$ (the number of symbols that can be avoided) using the condition in (12) as a guideline.

A close observation of Equation (12) shows that term q^{r-1} on the LHS counts all the r-tuples to be avoided even those having symbols from set A (see Table 4), while the RHS has already eliminated all control vectors produced by symbols in set A. This requires an adjustment of q^{r-1}, which takes into account the fact that the r-tuples to be avoided which have symbols from set A should not be counted. An adjustment in (12) results in the LHS becoming $|A|(n-k)(q-|A|)^{r-1}$, where we subtracted $|A|$ from q to yield $(q-|A|)^{r-1}$.

For brevity, let $X = |A|(n-k)$ and $Y = (q-|A|)$ in (12). It should be noted that the value of X is limited by q^{r-1} because there are exactly q^{r-1} unique combinations of $v_1v_2\ldots v_r$, after which there are repetitions.

Even with the adjustment in (12), of q^{r-1} to $(q-|A|)^{r-1}$, there are still two problematic cases not taken into account.

1. Not all the r-tuples to be avoided, which have symbols from set A, are always included in the adjustment of q^{r-1} to $(q-|A|)^{r-1}$. There still remains an unknown number of r-tuples to be avoided which have symbols from set A. We call this unknown number of the r-tuples to be avoided, the quantity λ, where $0 \leq \lambda \leq |A|X$.

2. There is an unknown number of r-tuples to be avoided that are repeated. The adjustment in (12) still assumes that all r-tuples to be avoided are unique, in the entire "space" XY^{r-1}. If there are repetitions, this "space" will definitely be reduced. To quantify the number of repetitions we define a quantity β, which gives us the number of repetitions.

It is, however, difficult, if not impossible, to precisely know the λ and β quantities. These quantities are not uniform across all codewords of a particular code. Moreover, these quantities will have to be used together taking into account overlaps. By overlaps we mean that an r-tuple(s) to be avoided can contain symbols from set A (adding to λ) and also be a repetition (adding to β).

Considering λ and β, the expression of the upper bound on $|A|$ is

$$|A|(n-k)(q-|A|)^{r-1} - \gamma < (q-|A|)^r. \tag{13}$$

The term on the LHS in (13), $\gamma\ (= \lambda + \beta)$, gives the sum of the quantities explained in cases 1 and 2, assuming no overlaps, $0 \leq \gamma < XY^{r-1}$.

Using Table 3 in Example 2, we present a pictorial explanation of the steps involved in arriving at Equation (13) (see Table 4). The table shows the three adjustments to (12) as described above. Firstly, a horizontal line is drawn to cut off the r-tuples that are precisely known to contain symbols in A, resulting to $(q-|A|)^{r-1} = 5$ rows. Next, we mark the remaining r-tuples containing symbols from set A with circles, and their total number is the quantity $\lambda = 18$. Finally, ignoring the r-tuples marked with circles, we mark repeated r-tuples with rectangles, and their total number is the quantity $\beta = 18$.

Table 4: Table listing all r-tuples to be avoided, for $n-k=4$, $r=2$ and $A=\{5,6,7\}$. $\lambda = 18$ and $\beta = 18$, resulting in $\gamma = 36$.

$P_1^C = 7$			$P_2^C = 4$			$P_3^C = 3$			$P_4^C = 7$		
$a_1 =$ 5	$a_1 =$ 6	$a_1 =$ 7	$a_1 =$ 5	$a_1 =$ 6	$a_1 =$ 7	$a_1 =$ 5	$a_1 =$ 6	$a_1 =$ 7	$a_1 =$ 5	$a_1 =$ 6	$a_1 =$ 7
$v_1\ v_2$	$v_1\ v_2$	$v_1\ v_2$	$v_1\ v_2$	$v_1\ v_2$	$v_1\ v_2$	$v_1\ v_2$	$v_1\ v_2$	$v_1\ v_2$	$v_1\ v_2$	$v_1\ v_2$	$v_1\ v_2$
$\enclose{circle}{6\ 0}$	3 0	0 0	1 0	2 0	$\boxed{3\ 0}$	$\boxed{1\ 0}$	4 0	$\enclose{circle}{7\ 0}$	$\boxed{3\ 0}$	$\boxed{4\ 0}$	$\boxed{0\ 0}$
1 1	4 1	$\enclose{circle}{7\ 1}$	0 1	3 1	2 1	$\enclose{circle}{5\ 1}$	$\boxed{0\ 1}$	$\boxed{3\ 1}$	$\boxed{1\ 1}$	$\enclose{circle}{6\ 1}$	$\boxed{2\ 1}$
3 2	$\enclose{circle}{6\ 2}$	$\enclose{circle}{5\ 2}$	$\boxed{3\ 2}$	0 2	1 2	2 2	$\enclose{circle}{7\ 2}$	4 2	$\enclose{circle}{7\ 2}$	$\boxed{0\ 2}$	$\boxed{4\ 2}$
4 3	1 3	2 3	$\boxed{2\ 3}$	$\boxed{1\ 3}$	0 3	$\enclose{circle}{6\ 3}$	3 3	$\boxed{0\ 3}$	$\enclose{circle}{5\ 3}$	$\boxed{2\ 3}$	$\enclose{circle}{6\ 3}$
$\enclose{circle}{7\ 4}$	2 4	1 4	$\enclose{circle}{5\ 4}$	$\enclose{circle}{6\ 4}$	$\enclose{circle}{7\ 4}$	$\enclose{circle}{7\ 4}$	$\boxed{2\ 4}$	$\boxed{1\ 4}$	0 4	$\enclose{circle}{7\ 4}$	3 4
0 5	5 5	6 5	4 5	7 5	6 5	3 5	6 5	5 5	2 5	5 5	1 5
2 6	7 6	4 6	7 6	4 6	5 6	4 6	1 6	2 6	4 6	3 6	7 6
5 7	0 7	3 7	6 7	5 7	4 7	0 7	5 7	6 7	6 7	1 7	5 7

$q^{r-1} = 8$

$(q-|A|)^{r-1} = 5$

$|A|(n-k) = 3\times4 = 12$

As it can be seen, Table 2 shows what goes on with just one codeword in which we want to avoid symbols 5, 6 and 7. Each codeword of a code, that has undesired symbols, when investigated this way will have its own value of γ, which may not be the same as that of another codeword. This presents a great difficulty in estimating the bound. To determine the bound in (13), one has to find the minimum value of γ among the codewords investigated.

3. ANALYSIS OF SIMULATION RESULTS

By performing an exhaustive computer search for the bound on $|A|$ in relation to r for several RS codes, we obtained the results in Tables 5, 6 and 7. The results show the limit to possible number of symbols that can be avoided, $|A|$.

Since the set A is a subset of GF(q) with cardinality $|A|$, where $|A|$ is the number of symbols to be avoided, there are $\binom{q}{|A|}$ possible sets with this cardinality. In Tables 5, 6 and 7 we list the bounds on $|A|$, for the cases when all the $\binom{q}{|A|}$ sets reach the same upper bound.

From the Tables 5, 6 and 7, we learn the following. For $k \leq (n-k)$, we can avoid up to n symbols when $r = k-1$ (remember that $k = r+k'$, then $r = k-1$ means $k' = 1$). However, avoiding n symbols will leave only one codeword in the codebook of the RS code, which is not a meaningful code. So we shall limit the avoidance of symbols to $n-1$, which means only a minimum of two symbols will be remaining in the alphabet since $n=q-1$. Since the results show that up to n symbols can be avoided for $r=k-1$, it is obvious that $n-1$ symbols can also be avoided.

Table 5: The maximum number of symbols $|A|$ that can be avoided for each r and k for RS codes over GF(7). Symbol avoidance is possible for all the $\binom{q}{|A|}$ sets.

$k =$	2	3	4
$(n-k) =$	4	3	2
$r = 1$	$\|A\| \leq 6$	$\|A\| \leq 2$	$\|A\| \leq 2$
$r = 2$		$\|A\| \leq 6$	$\|A\| \leq 6$

Table 6: The maximum number of symbols $|A|$ that can be avoided for each r and k for RS codes over GF(11). Symbol avoidance is possible for all the $\binom{q}{|A|}$ sets.

$k =$	2	3	4	5	6
$(n-k) =$	8	7	6	5	4
$r = 1$	$\|A\| \leq 10$	$\|A\| \leq 1$	$\|A\| \leq 1$	$\|A\| \leq 1$	$\|A\| \leq 2$
$r = 2$		$\|A\| \leq 10$	$\|A\| \leq 3$	$\|A\| \leq 4$	$\|A\| \leq 4$
$r = 3$			$\|A\| \leq 10$	$\|A\| \leq 5$	$\|A\| \leq 5$
$r = 4$				$\|A\| \leq 10$	$\|A\| \leq 10$

4. DECODING

We now know that the code $(n,\ k',\ d')W'$ over GF(q') is produced by the symbol avoidance operation from the linear RS code $(n,\ k,\ d)W$ over GF(q). The decoding algorithm of the RS code W is known and well defined. The code W' is easily decoded using the decoding algorithm of the code W, but with some "minor added operations" which enhance the performance of W'. To describe these minor added operations, let $D \in W'$, and $\tilde{D}$ be a received noise corrupted codeword version of D. We assume additive noise which changes one symbol into another within the GF(q). Note that $\tilde{D}$ can have undesired symbols due to noise corruption in the channel.

Decoding steps of $\tilde{D}$:

1. If the received codeword $\tilde{D}$ has undesired symbols, the undesired symbols are marked as erasures and then minimum distance decoding is performed.
2. If the minimum distance decoding results in a codeword not in W', an error is detected and the codeword is decoded to the nearest codeword in W'.

Table 7: The maximum number of symbols $|A|$ that can be avoided for each r and k for RS codes over GF(13). Symbol avoidance is possible for all the $\binom{q}{|A|}$ sets.

$k =$	2	3	4	5	6
$(n-k) =$	10	9	8	7	6
$r = 1$	$\|A\| \leq 12$	$\|A\| \leq 2$	$\|A\| \leq 1$	$\|A\| \leq 1$	$\|A\| \leq 1$
$r = 2$		$\|A\| \leq 12$	$\|A\| \leq 3$	$\|A\| \leq 3$	$\|A\| \leq 4$
$r = 3$			$\|A\| \leq 12$	$\|A\| \leq 5$	$\|A\| \leq 5$
$r = 4$				$\|A\| \leq 12$	$\|A\| \leq 7$
$r = 5$					$\|A\| \leq 12$

The code W' is highly likely to have better performance than the original RS code W because of one or a combination of the following:

- $d' \geq d$.
- In some cases, the weight enumerator of W' is improved compared to that of W. This is because some codewords of W are excluded in W'.

5. CONCLUSION

A procedure for avoiding symbols in a Reed-Solomon code was presented. The procedure created a new Reed-Solomon code (W') from an original Reed-Solomon code (W), but linearity cannot be guaranteed in the new code. An analysis on how to obtain an upper bound on the number of symbols that can be avoided in the new Reed-Solomon code was presented. Unfortunately, a final analytic expression for the upper bound was not found. The expression we came up with could only serve as guidelines towards the upper bound. We resorted to computer searches and found numerous results for the upper bound on the number of symbols that can be avoided in W'. A decoding procedure for the code W' was described.

REFERENCES

[1] G. Solomon, "A note on alphabet codes and fields of computation," *Information and Control*, vol. 25, no. 4, pp. 395–398, Aug. 1974.

[2] B. Sklar, *Digital Communications: Fundamentals and Applications*. Prentice Hall Inc., 1988.

[3] S. Lin and D. J. Costello Jr., *Error control coding: Fundamentals and Applications*. Prentice Hall Inc., 1983.

[4] G. Solomon, "Non-linear, non-binary cyclic group codes," in *Proceedings of the 1993 IEEE International*

Symposium on Information Theory, San Antonio, TX, USA, 1993, pp. 192–192.

[5] M. Hattori, R. J. McEliece, and G. Solomon, "Subspace subcodes of Reed-Solomon codes," *IEEE Transactions on Information Theory*, vol. 44, no. 5, pp. 1861–1880, 1998.

[6] A. Urivskiy, "On subset subcodes of linear codes," in *Problems of Redundancy in Information and Control Systems (RED), 2012 XIII International Symposium on*, Moscow, Russia, 2012, pp. 89–92.