A Proposed Algorithm for Generating the Reed-Solomon Encoding Polynomial Coefficients over GF(256) for RS[255,223]8,32

Frimpong Twum
Department Of Computer
Science Kwame Nkrumah
University of Science and Technology,
Kumasi, Ghana

J. B. Hayfron-Acquah
Department Of
Computer Science
Kwame Nkrumah University of
Science and Technology,
Kumasi, Ghana

W. W. Oblitey
Department Of
ComputeScience Indiana
University of Pennsylvania,USA.

R. K. Boadi
Department Of Mathematics
Kwame Nkrumah
University of
Science and Technology,
Kumasi, Ghana.

ABSTRACT
The ability to detect and correct data loss is of crucial importance to securing and recovering data stored on any storage facility (most importantly, the cloud). Reed-Solomon (RS) codeword is the most used for achieving this purpose. RS codeword is widely used for detecting and recovering data transmission errors as well as data loss in storage. This paper illustrates how the coefficients of the encoding polynomial needed for the generation of the RS codeword are generated. An efficient algorithm for generating the encoding polynomial coefficient is proposed. The algorithm is implemented in JAVA for Galois Field [GF(256)] with 32 parity shards – RS[255,223]8,32 to obtain an array of 32 coefficients as follows: {232, 29,189, 50, 142, 246, 232, 15, 43, 82, 164, 238, 1, 158, 13, 119, 158, 224, 134, 227, 210, 163, 50, 107, 40, 27, 104, 253, 24, 239, 216,45}

Keywords
Reed Solomon Codes, Galois Field, Encoding Polynomial, Error detection and Correction

1. INTRODUCTION
The application of finite field (Galois Field) over the last few decades has been enormous especially in the areas of data communication and storage [1], [2]. Other usages have been as follows: encryption, and data compression. Reed Solomon (RS) codes which operates over Galois Fields has been used extensively for the detection and correction of errors that occurs during data transmission and data storage [3]. A study by Ref. [4] outline other application areas for RS codes as Voyager spacecraft, detecting and correcting data losses in wireless transmission, dealing with scratches on CD's, correcting scanning errors in QR codes among others. This paper presents a concise approach for generating the encoding polynomial used for the generation of the RS codeword that is used for the detection and correction of data transmission errors and data storage losses. The study also proposes an efficient algorithm for generating the encoding polynomial for the generation of the RS codeword based on GF(256) for RS[255,223]8,32.

2. RELATED WORKS
WHAT IS A FIELD?

A set of numbers is a field if it satisfies the following properties.

Field Properties
The real number system is primarily a set, for e.g. {a, b, c, …}, on which the operations of addition and multiplication are defined in such a way that for all pair of real numbers there exist a unique sum and product that are also real numbers and thus exhibit properties as follows [5], [6].

1. **Cummulative Laws**
   a + b = b + a ---------- addition.

E.g. 1 + 2 = 2 + 1.

ab = ba ---------- multiplication.

E.g. 1 * 2 = 2 * 1.

2. **Associative Laws**
   (a + b) + c = a + (b + c) ---------- addition.

E.g. (1 + 2) + 3 = 1 + (2 + 3).

(ab) c = a (bc)----------multiplication. E.g. (1 * 2) * 3 = 1 * (2 * 3).

3. **Distributive Laws**
   a (b + c) = ab + ac

E.g. 1 (2 + 3) = (1*2) + (1*3)

There are distinct real numbers 0 and 1 such that

a + 0 = a and a * 1 = a

NB: For addition, the identity is '0'. Whereas multiplication identity is 1.

1. For each 'a' there is a real number ' -a ' such that a + (-a) = 0 and if a ≠ 0 there is a real number

$\frac{1}{a}$ (or a^{-1}) such that

a ($\frac{1}{a}$) = 1

2. a + b ∈ R, and a*b ∈ R (closure laws),
   Example

   For a given set to be a field it should satisfy all the field properties.

   An a example of a field is the set of rational numbers.

   i.e.

   a + b ∈ Q and a*b ∈ Q .

   e.g. if a = 2

   a + (-a) = 0 and

   a ($\frac{1}{a}$) = 1

What is A Finite Field?
A finite field (aka. Galois Field – GF) is a field with finitely defined elements where upon performing the arithmetic operations of addition, subtraction, division, or multiplication of f(p) mod p on any two of the field elements, the result is always an element in the set.

F_p = {0, 1, 2, …, p-1}

•, +: integer addition and multiplication in modulo p

This property of a finite field enables its usage for error detection and data recovery in data communication and data storage [5], [6].

A finite field is constructed using a **prime number base** or **powers of a prime number**. This is to ensure a unique value is obtained when addition and multiplication operations are performed on any two of the field elements. For example, finite field elements for GF(2) is constructed as {0, 1}, GF(3) as {0, 1, 2}, and GF(7) as {0, 1, 2, 3, 4, 5, 6} [7].

In the case of the powers of prime, a finite field for GF(2^n), where 2 is the prime base and 'n' is the exponent determines the number of elements in the field.

As an example GF(2^3) which is the same as GF(8) has field elements as {0, 1, 2, 3, 4, 5, 6, 7}, and GF(16) represented in prime powers of 2 as GF(2^4) has elements as {0, 1, 2, 3, 4, 5, 6, 7, 8, 9,0, 11, 12, 13, 14, 15}. It follows therefore that GF(N) = GF(2^n) has field elements as {0, 1, 2, …, n-1). Hence GF(2^8) = GF(256) has 256 field elements as {0, 1, 2, 3, …, 255}. This is an example of a modulus 256 field and hence 255 is the maximum value [2], [7], [8].

For computer computational operations, a base 2 prime base is used for representing the field elements as prime powers [4].

The elements of a finite field are usually represented as polynomials that take their coefficients from a particular field F_p. For example, for a polynomial, $F_p(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$ where $a_i \in F$. A deg.1 polynomial of $F_2(x) = a_0 + a_1x$, has field elements represented using alpha powers as {α, 1+ α} which are already in their irreducible form. Elements of deg.2 polynomials in Fp(x) = $a_0 + a_1x + a_2x^2$ are obtained as {α^2, α^2+ α, α^2+1, α^2+ α+1} [2], [9].

Similarly for deg.3 polynomials in Fp(x) = $a_0 + a_1x + a_2x^2 + a_3x^3$ the elements are as follows {α^3, α^3+ α^2, α^3 +α^2+ α, α^3+ 1, α^3 +α^2+ 1 , α^3+ α+1, α^3+ α, α^3+ α^2 +α+1}.

Galois Field Elements Construction
Galois field represented as binary form is very convenient for detecting and correcting errors (in transmission or storage) and as well as for ciphering computer data. This is because it is a finite field and adheres to properties of a field. The elements of Galois field, GF(P^m) is defined as

$F_p^m = \{a_0 + a_1x + a_2x^2 + \dots + a_{m-1}x^{m-1}\}$ where $a_i \in F_p$.

+ : addition in $F_p(x)$ mod p

• : multiplication in $F_p(x)$ mod π(x)

Where π(x):= deg.m irreducible polynomial in $F_p(x)$.

Irreducible polynomials (polynomial that cannot be factord)for example x^2+1 has no roots and are used to construct the elements of GF(2^n). Reducible polynomials for example, x^2-1, has roots as -1 and +1 and hence are not used when generating the elements of GF(2^n). As an example, given $F_p = F_4 = F_2^2$, the elements of polynomials of deg ≤ m-1 with coefficients from F_p are given as $F_p^m = F_2^2$ = {0, 1, α, 1+ α}, p=2, and m=2 [8], [9]. For F_2^3 the field elements in powers of alpha are obtained as follows:

deg. 0	deg.1	deg. 2
a_0	$a_0 + a_1\alpha$	$a_0 + a_1\alpha + a_2\alpha^2$

$F_p^m = F_2^3$ = {0, 1, α, α+1, α^2, α^2 + α, α^2+1, α^2+α+1}, where p=2, and m=3.

Galois Field (Gf) Arithmetics
Arithmetic in GF or finite field is different from standard arithmetic. Unlike standard arithmetic, which has an infinite number of elements, there is limited number of elements in a finite field.

Thus, arithmetic in finite field is basically carried out on a set of elements in which when the arithmetic operations of addition multiplication subtraction, or division is performed on the set the results is always found in the same set [9], [10].

Recall, a finite field of elements P^n is basically represented in Galois field as GF(P^n), where P is a prime base and n is the exponent of P, modulus P. For example, GF(8) = GF(2^3) modulus 8 and the elements in the field are {0, 1, 2, 3, 4, 5, 6, 7}.

Addition and Subtraction In Gf(8)
The steps to performing addition and subtraction in GF(2^3) are as follows:

- the polynomials of deg ≤ m-1 with coefficients from F_2^3, where m=3 and p=2 is defined to obtain the field elements for F_2^3as in section 2.3 above as: Elements of F_2^3 = {0, 1, α, α+1, α^2, α^2+ α, α^2+1, α^2+ α+1}
- the addition table is constructed using the resulting elements of F_2^3 in modulus 2 as follows:

The GF addition table of Table 1 also has the entries for the GF subtraction operation as subtraction is performed as addition in computer systems using the additive inverse of the subtrahend. An element's additive inverse is the element that results with zero when added to the minuend. **Rule:** a + (-a) = 0

Multiplication & Division In Gf(8)

- The multiplication table for GF(8) mod 2 is constructed using the elements of F_2^3 and a deg.3 irreducible primitive polynomial in F_2^3 obtained as

$\alpha^3+\alpha+1$ or $\alpha^3+\alpha^2+1$. This resulted with the field elements for $F_2{}^3$ as shown in powers of alpha and is used for the construction of the multiplication table, Table 2.

$F_2{}^3 = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$, mod 2

Using the irreducible polynomial $\alpha^3+\alpha+1$ $\alpha^3 = \alpha + 1\alpha^4 = \alpha(\alpha^3) = \alpha(\alpha + 1) = \alpha^2 + \alpha$

$\alpha^5 = \alpha(\alpha^4) = \alpha(\alpha^2 + \alpha) = \alpha^3 + \alpha^2$, Now Substituting for $\alpha^3 = \alpha + 1$ gives

$\alpha^5 = \alpha^2 + \alpha + 1$

$\alpha^6 = \alpha(\alpha^5) = \alpha(\alpha^2 + \alpha + 1) = \alpha^3 + \alpha^2 + \alpha$, Now Substituting for α^3 gives $(\alpha + 1 + \alpha^2 + \alpha) \bmod 2$ $\therefore \alpha^6 = \alpha^2 + 1$

$\alpha^7 = \alpha(\alpha^6) = \alpha(\alpha^2 + 1) = \alpha^3 + \alpha$ Substituting for α^3 gives $(\alpha + 1 + \alpha) \bmod 2 = 1$

$\alpha^8 = \alpha(\alpha^7) = \alpha(1) = \alpha$, $\alpha^9 = \alpha^2$, $\alpha^{10} = \alpha^3$, $\alpha^{11} = \alpha^4$, $\alpha^{12} = \alpha^5$, $\alpha^{13} = \alpha^6$

As can be seen, the element values for alpha repeats from α^7 indicating GF(8) is a field.

Table 2 also present entry values for division in GF(8) which is performed using multiplicative inverse of the elements in the set. **Rule: $a * (a^{-1}) = 1$**

As an example dividing α^5 by α^3 or (7/3) imply multiplying α^5 by the multiplication inverse of α^3 as follows: From Table 2, the multiplication inverse of α^3 is the corresponding element in the matrix that when multiplied by α^3 gives 1 as the result (i.e. **Rule: $a * (a^{-1}) = 1$**).

Hence the inverse of α^3 is $\alpha^4 = 6$. Therefore α^5 / α^3 is obtained ($\alpha^5 * \alpha^4 = \alpha^2$). Thus: 7/3 implies 7 * 6 = 4 [where 6 is the inverse of 3].

It can be seen from the Table 1 and Table 2 there are no identical entry in any of the rows or columns and there are also no repeating or negative entries in any row or column. These characteristics of the field element set makes the use of Galois field ideal for data recovery and/or error detection in data communication and/or data storage. Any of the elements in the set can be regenerated from the rest of the elements in the event of loss or damage. This is useful particularly in distributed data storage as cloud computing as in the event of a system breakdown or disk drives failures, the system can recover missing data and prevent any data loss. This system of data recovery is much efficient and cost effective than those of RAID technology [10].

The RS Codeword

The ability to detect and correct data loss is of crucial importance to security and recovering data stored on any storage facility (most importantly, the cloud). Reed-Solomon (RS) codeword is the most used for achieving this purpose. The following section illustrates how the RS codewords are generated and used for the detection and correction of errors in data transmission and storage.

According to Ref. [3], [9] the RS codeword is generated using three (3) polynomials namely:

The "**Irreducible Polynomial**" (i.e. the polynomial equivalent of a prime number) is used as the generating polynomial for the Galois field elements generation. For the GF (8) elements generation, the irreducible polynomial (**$\alpha^3+\alpha+1$**) is used.

The "**Generator Polynomial**" – This polynomial is required for generating the encoding polynomial (which is the 3rd polynomial needed for the generation of the RS codeword). The generator polynomial is a generic polynomial of the form; **$G(x) = (x-\alpha^1)(x-\alpha^2)(x-\alpha^3)\ldots(x-\alpha^{2t})$,** where α^1, α^2, α^3 , etc. are the field elements and the value 2t determines the number of the Forward Error Correction (FEC) require.

For example, assuming an RS codeword of **RS [7,5] S=3, t=2** where 'S' is the number of bits making a symbol size (in this case 3-Bit symbols), 't' is the number of the 3-Bits symbols used for error correction (in this case 2 (3-Bits) symbols), and 5 is the number of 3-Bits symbols used for representing the actual data chunks, whiles 7 is the total number of RS codewords for a GF (8). The generic expression (a.k.a. Maximum Distance Separable-MDS) for RS codeword is given as **RS [n, k] s, t,** where n is the number of codewords given as $2^s - 1$ and k is the number of data chunks.

Encoding Polynomials – for the RS[7,5]3,2 specification codeword example, 2 symbols are needed for FEC and hence only 2 of the Generator Polynomials are required as follows: **$G(x) = (x-\alpha^1)(x-\alpha^2)$.**

Since addition and subtraction operations give the same result in GF, **$G(x) = (x+\alpha^1)(x+\alpha^2)$.** From the Table 1, **$\alpha^1 = 2$,** and **$\alpha^2 = 4$** therefore **$G(x) = (x+2)(x+4)$.** Hence**, $G(x) = x^2 + 4x + 2x + 8 = x^2 + 6x + 8$** 8 in binary is 1000 which is bigger than the largest field elements of 7 therefore the Generating polynomial of 1011 is XOR.

```
   1000
XOR 1011
    0011 = 3.
```

Therefore, **$G(x) = x^2 + 6x + 3$** is the encoding polynomial which is expressed also as 163 and is used for the RS codeword generation.

3. METHODOLOGY

GENERATION OF COEFFICIENT VALUES OF THE ENCODING POLYNOMIAL

This study is aimed at proposing an efficient algorithm for the generation of the encoding polynomial coefficients for GF(256) that are needed for the generation of the RS codeword of RS[255,223]8,32. To achieve this aim require 3 polynomials as follows:

- The irreducible polynomial (also referred to as the generating polynomial)
- The generator polynomial
- The encoding polynomial

As noted in the case of GF(8) earlier in the literature review, for GF(256) or GF(2^8) (the focus of this study), the number of RS codewords generated is obtained as $n = 2^8-1 = 255$. Hence for the 32 (8-bits Symbols) parity shards or (32 Forward Error Correction (FEC) codes) require by this study imply splitting files into 223 (8-bits symbols) data shards. To achieve this, the following are undertaken.

Step-1: A degree 8 irreducible polynomial (a polynomial equivalent of a prime number) in $F_2{}^8$ obtained as $P(x) = \alpha^8+\alpha^4+\alpha^3+\alpha^2+1 = 285$ [2], [11], is used to generate the GF(256) field elements of (0-255).

Step-2: The generator polynomial which is needed for the generation of the encoding polynomial is defined for the

creation of 32 (8-bits Symbols) parity shards or (32 Forward Error Correction (FEC) codes) as follows: **G(x) = (x-α^1) (x-α^2) (x-α^3) (x-α^4) (x-α^5) (x-α^6) (x-α^7) (x-α^8) (x-α^9) (x-α^{10}) (x-α^{11}) (x-α^{12}) (x-α^{13}) (x-α^{14})(x-α^{15}) (x-α^{16}) (x-α^{17}) (x-α^{18}) (x-α^{19}) (x-α^{20}) (x-α^{21}) (x-α^{22}) (x-α^{23}) (x-α^{24}) (x-α^{25}) (x-α^{26}) (x-α^{27})(x-α^{28}) (x-α^{29}) (x-α^{30}) (x-α^{31}) (x-α^{32}) ----------[expression 1]**

Now substituting values of '**α**' in Step-1 with their decimal equivalent from the GF(256) elements

G(x) = (x+2) (x+4) (x+8) (x+16) (x+32) (x+64) (x+128) (x+29) (x+58) (x+116) (x+232) (x+205) (x+135) (x+19) (x+38) (x+76) (x+152) (x+45) (x+90) (x+180) (x+117) (x+234)(x+201) (x+143) (x+3) (x+6) (x+12) (x+24) (x+48) (x+96) (x+192) (x+157) ----------[expression 2]

Since the addition and subtraction arithmetic operation in GF gives the same results, the subtraction operator in [expression 1] is replaced with addition in [expression 2].

Step-3: The generator polynomial of [expression 2] is then expressed in the form

$G(x) = a_{32}x^{32}+ a_{31}x^{31}+ a_{30}x^{30}+ a_{29}x^{29}+ a_{28}x^{28}+ a_{27}x^{27}+ a_{26}x^{26}+ a_{25}x^{25}+ a_{24}x^{24}+ a_{23}x^{23}+ a_{22}x^{22}+ a_{21}x^{21}+ a_{20}x^{20}+ a_{19}x^{19}+ a_{18}x^{18}+ a_{17}x^{17}+ a_{16}x^{16}+ a_{15}x^{15} +a_{14}x^{14}+ a_{13}x^{13}+ a_{12}x^{12}+ a_{11}x^{11}+ a_{10}x^{10}+ a_9x^9+ a_8x^8+ a_7x^7+ a_6x^6+ a_5x^5+ a_4x^4+ a_3x^3+ a_2x^2+ a_1x^1$ ----------**[expression 3]**

Where the coefficient values ($a_{32},a_{31},a_{30},a_{29},a_{28},a_{27},a_{26},a_{25},a_{24},a_{23},a_{22},a_{21},a_{20},a_{19},a_{18},a_{17},a_{16},a_{15},a_{14},a_{13},a_{12},a_{11},a_{10},a_9,a_8,a_7,a_6,a_5,a_4,a_3,a_2,a_1x^1$) are used for the generation of the encoding polynomial which is used for the generation of the RS codeword for error detection and recovery in the vent of data loss, damage, or alteration in transmission or in storage. The algorithm proposed by this study for the generation of the coefficient values is as explained below.

ALGORITHM FOR THE GENERATION OF THE COEFFICIENTS OF THE ENCODING POLYNOMIAL

The Generator polynomial is of the form

$$g(x) = (x + \alpha^1)(x + \alpha^2)(x + \alpha^3) \dots (x + \alpha^n)$$

where n is the number of parity shards

The coefficients of x in the expansion of $g(x)$ is found using the algorithm below

for pow = 0 to n

→ sum all the combinations of the set $\{\alpha^1, \alpha^2, \alpha^3 \dots \alpha^n\}$

that have pow elements

For example, to find the coefficients for $n = 4$, assuming the Galois Field elements are

- $\alpha^1 = a$
- $\alpha^2 = b$
- $\alpha^3 = c$
- $\alpha^4 = d$

Then the looping process perform the following action

- $pow = 0: \{\}$
- $pow = 1: a + b + c + d$
- $pow = 2: ab + ac + ad + bc + bd + cd$
- $pow = 3: abc + abd + acd + bcd$

Thus

1. *Loop* $n + 1$ *times* $(i \rightarrow 0$ *to* $n)$
   a. *Generate all subsets of the alpha exponents set, of size* $(n - i)$
   b. *Multiply the components of each of the subsets*
   c. *Add the products of the subset components*
   d. *Save the sum of the subsets in a coefficients array*

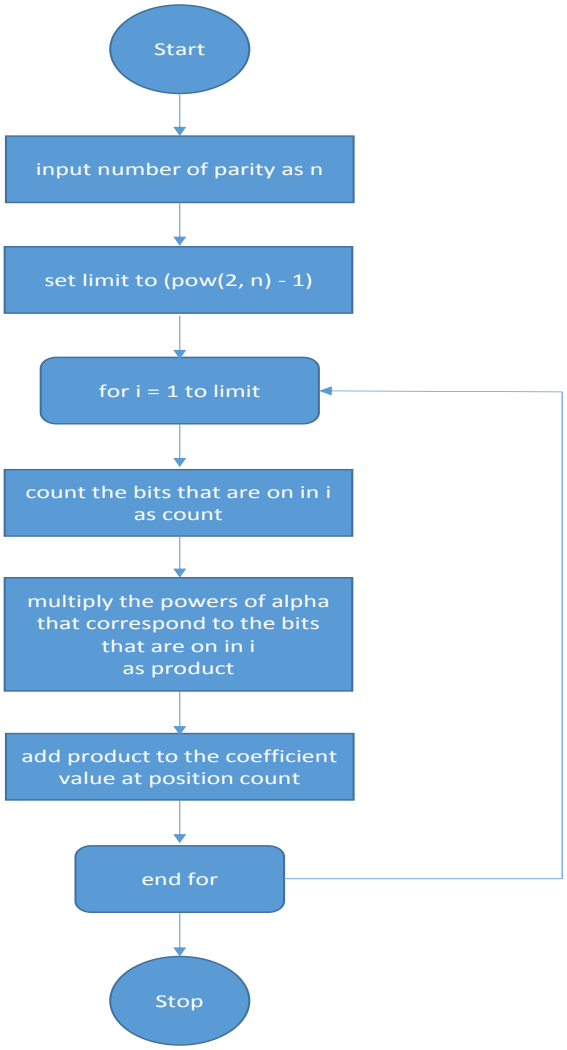

Figure 1: Flow diagram for the algorithm

4. IMPLEMENTATION

GENERATING THE GF(256) FIELD ELEMENTS

The process of representing a finite field in a computer, especially for arithmetic purposes has been refined. Representing the elements of GF(8) like this {0, 1, 2, 3, 4, 5, 6, 7} for example is more difficult to implement than like this {0, 1, 2, 4, 3, 6, 7, 5}. Doing arithmetic in GF by hand is not much of a problem. However by using a computer the elements of the field are best represented as exponents of 2. An irreducible polynomial is used as a modulus to ensure the exponents of 2 do not repeat.

This example demonstrates the use of the irreducible polynomial 29(i.e. 285-256) to keep the powers of 2 within the range 0 - 255

** a^0 = 2^0 = 1

** a^1 = 1 * 2 = 2

** a^2 = 2 * 2 = 4

** a^3 = 4 * 2 = 8

** a^4 = 8 * 2 = 16

** a^5 = 16 * 2 = 32

** a^6 = 32 * 2 = 64

** a^7 = 64 * 2 = 128

** a^8 = 128 * 2 = 256

Since 256 is outside the range 0 - 255, subtract 256 from it to put it back within range, then XOR with the irreducible polynomial (29) to get a unique start value.

Hence a^8 = (256 - 256) ⊕ 29 = 29

** a^9 = 29 * 2 = 58

** a^10 = 58 * 2 = 116

** a^11 = 116 * 2 = 232

** a^12 = 232 * 2 = 464

Once again, 464 is outside the range so subtract 256 to put it back within range, then add 29 to get a unique number. Hence the calculation continues as

∴ a^12 = (464 - 256) ⊕ 29 = 205

464	
-256	
208 =	11010000
⊕29=	11101
	11001101 = 205

** a^13 = 205 * 2 = 410 === (410 - 256) ⊕ 29 = 135

** a^14 = 135 * 2 = 270 === (270 - 256) ⊕ 29 = 19

** a^15 = 19 * 2 = 38

** a^16 = 38 * 2 = 76

As can be seen, by repeatedly subtracting 256 from any product value that falls outside of the range of 0-255 and XOR with the irreducible polynomial of 29 the process keeps generating unique numbers within the 0 -255 range resulting with the GF(256) table (See Table 3).

IMPLEMENTING ARITHMETIC OPERATION IN GF
ADDITION AND SUBTRACTION IN GF

Additions and subtractions in a Galois Field both come down to the bitwise XOR operation because in GF(2)

a. 1 + 1 = 0 --> 0 - 1 = 1

b. 1 + 0 = 1 --> 1 - 1 = 0 AND 1 - 0 = 1

c. 0 + 0 = 0 --> 0 - 0 = 0

Per the definition of the XOR operation, the result is 0 when the operands are alike and 1 when the operands are different. That definition is satisfied by both addition and subtraction in GF(2).

Thus the implementation of the addition and subtraction functions in Java code is simply to XOR the arguments.

MULTIPLICATION AND DIVISION IN GF

Multiplication contains an element of addition in it (in fact, multiplication is simply repeated addition) but since addition is implemented as an XOR operation in GF(2), repeated addition will always result in an answer of 0.

An alternative is to use this addition feature of logarithms

A * B = log-inverse(log(A) + log(B));

Division in the Galois Field is also implemented in Java using logarithms

A / B = log-inverse(log(A) - log(B));

LOGARITHMS and EXPONENTS

The precondition for using the addition and subtraction features of logarithms in the multiplication and division in Galois Fields is that the log values for all the elements of the field must first be known. Fortunately the method used earlier to generate the elements of the Galois Field uses exponents of 2.

Thus the log of any element in the field is the exponent of 2 (or exponent $\boldsymbol{\alpha}$ as in Table 1) indexed to the position corresponding to its generated exponent value. For example, 2^3 generated 8, so log(8) = 3, and 2^7 generated 128, and hence log(128) = 7, in GF(256).

IMPLEMENTING THE ALGORITHM FOR THE GENERATION OF COEFFICIENTS OF THE ENCODING POLYNOMIAL GENERATION ALGORITHM

Both the encoding and decoding processes of the Reed-Solomon algorithm rely on a generator polynomial of the form

$$g(x) = \prod_{i=1}^{n} (x - \alpha^i)$$

where n is the number of parity shards.

It should be noted that with Galois Field arithmetic, addition and subtraction both result in the same XOR operation. As such, the algorithmic representation of the above formula used the Galois Addition method.

It was determined that the coefficients resulting from the expansion of the encoding polynomial formula

$$g(x) = \sum_{i=0}^{n} c_i x^{n-i}$$

followed this pattern

$$c_i = \sum_{j=1}^{\binom{2t}{j}} \prod k$$

where k is a member of one subset of the Galois Field's elements that has i members

The algorithm focused on determining the coefficients only. This involved finding all the possible combinations of the first 2t Galois Field Elements and performing the appropriate multiplication and addition operations on them. The process used is described in greater detail below:

1. Obtain the first $2t$ values in the Galois field and save them in an array
2. Determine the highest integer (upper limit) that has $2t$ bits using the formula below

$$limit = 2^{2t} - 1$$

3. Create $2t$ masks for determining which bits are on in any given integer that is within the range $[0, limit]$.The masks are simply integer values whose bit representations have only one bit on and all other bits off. In other words, the masks are the powers of 2 from 2^0 to 2^{2t}. When a bitwise AND operation is performed with a number and any of the masks, a result of 0 means that the particular bit which the mask has on, is off in the number. However, a result greater than 0 indicates that that particular bit is on. Using this approach, it is possible both to determine which particular bits are on in the number and also count them.
4. Create $2t$ accumulators, initialized to 0. These accumulators will hold the values of the coefficients when the program runs to a completion.
5. The combinations of the first 2t Galois Field elements are generated as integer values from 1 to $limit$. Thus, a loop is used which runs from 1 to $limit$. For each integer in the range
    a. The masks are applied to the integer to determine which bits are on.
    b. The Galois Field elements whose positions corresponds to the bits that are on in the integer, are multiplied to get the product of all the elements in that particular combination.
    c. Concurrently, the number of bits that are on are counted. The product from multiplying the bits that are on, is added to the accumulator at the position of the count. This ensures that all the products derived from multiplying combinations of a particular length are accumulated in a single accumulator.

PSEUDOCODE FOR THE COEFFICIENT GENERATION ALGORITHM

1. $Iimit = 2^n - 1$
2. $for\ i = 1\ to\ limit$
    a. $product = 1$
    b. $for\ j = 1\ to\ n$
        i. $x = i\ XOR\ mask[j]$
        ii. $if\ x > 1$
            1. $product = product * \exp[j]$
            2. $count = count + 1$
    c. $coefficient[count] = coefficient[count] + product$

- The array $mask[]$ contains the powers of two from 1 to 2^n
- The array $\exp[]$ contains the Galois field elements
- The array $coefficient[]$ holds the values of the coefficients at the end of the program execution

5. RESULTS

This algorithm was applied to $GF(256, 285)$ with $n = 32$.

The resulting values of $coefficient[]$ obtained from a Java code implementation are as follows:

```
4294967291
4294967292
4294967293
4294967294
4294967295
232  29  189  50  142  246  232  15  43  82  164  238  1  158  13  119  158  224  134  227  210  163  50  107  40  27  104  253  24  239  216  45
Process finished with exit code 0
```

6. CONCLUSION

This research provides overview of the Galois field theory and Reed Solomon Coding. It proposes and implements an algorithm for the generation of the encoding polynomial coefficients that are used for the Reed Solomon codeword generation. The outcome of this study enables the generation of the Reed Solomon codeword that is used for detection and correction of data loss in transmission or storage.

7. REFERENCES

[1] Plank, J. S. (2013). Erasure Codes for Storage Systems. Available at: https://www.usenix.org/system/files/login/articles/10_plank-online.pdf

[2] REDTITAN, (2011). Error detection and correction. Available at: http://www.pclviewer.com/rs2/galois.html

[3] cs.cmu.edu (1998). Reed-Solomon Codes. An introduction to Reed-Solomon codes: principles, architecture and implementation. Available at: https://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed_solomon_codes.html

[4] Cox, R. (2012). Finite Field Arithmetic and Reed-Solomon Coding. Available at: http://research.swtch.com/field

[5] Trench W. F., (2003). *Introduction to Real Analysis*. Library of Congress Cataloging-in-Publication Data. Available at: http://ramanujan.math.trinity.edu/wtrench/texts/TRENCH_REAL_ANALYSIS.PDF

[6] Wang, J. (2009). *Computer Network Security Theory and Practice*. Springer

[7] Benvenuto, C. J. (2012). Galois Field in Cryptography. Available at: https://www.math.washington.edu/~morrow/336_12/papers/juan.pdf

[8] Lynson, B. Tutorial on Reed-Solomon Error Correction Coding. NASA Tech Brief MSC-21834. Available at: http://jeffareid.net/misc/msc-21834.pdf

[9] Hill, T. (2013). Reed Solomon Codes Explained. Available at: https://www.tony-**hill**.info/app/download/.../**Reed**+**Solomon**+**Explained**+V1-0.pdf

[10] Plank, J. S. (1997). A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems. Software-Practice and Experience. Vol.27, No.9, 995–1012. Available at: http://cgi.di.uoa.gr/~ad/M155/Papers/RS-Tutorial.pdf

[11] Mathematics Stack Exchange, (2011). Addition and Multiplication in a Galois Field Available at: *http://math.stackexchange.com/questions/89805/addition-and-multiplication-in-a-galois-field*

8. APPENDIX

Table 1: Addition in GF(8) mod 2

+	**0**	**1**	**α =2**	α^3 **α + 1=3**	α^2 **=4**	α^4 **α^2 + α =6**	α^6 **α^2 + 1=5**	α^5 **α^2 + α + 1=7**
0	0	1	2	3	4	6	5	7
1	1	0	3	2	5	7	4	6
α =2	2	3	0	1	6	4	7	5
α^3 **α + 1=3**	3	2	1	0	7	5	6	4
α^2 **=4**	4	3	6	7	0	2	1	3
α^4 **α^2 + α=6**	6	7	4	5	2	0	3	1
α^6 **α^2 + 1=5**	5	4	7	6	1	3	0	2
α^5 **α^2 + α + 1=7**	7	6	5	4	3	1	2	0

Table 2: Multiplication in GF(8) mod 2

*	**0**	α^7 **= 1**	**α =2**	α^2 **=4**	α^3 **= α + 1 =3**	α^4 **=α^2 + α =6**	α^5 **= α^2 + α + 1 =7**	α^6 **= α^2 + 1 =5**
0	0	0	0	0	0	0	0	0
α^7 **= 1**	0	α^7 = 1	α=2	α^2=4	α^3=3	α^4=6	α^5=7	α^6=5
α =2	0	α=2	α^2=4	α^3=3	α^4=6	α^5=7	α^6=5	α^7 = 1
α^2 **=4**	0	α^2=4	α^3=3	α^4=6	α^5=7	α^6=5	α^7 = 1	α=2
α^3 **= α + 1 =3**	0	α^3=3	α^4=6	α^5=7	α^6=5	α^7 = 1	α=2	α^2=4
α^4 **=α^2 + α =6**	0	α^4=6	α^5=7	α^6=5	α^7 = 1	α=2	α^2=4	α^3=3
α^5 **= α^2 + α + 1 =7**	0	α^5=7	α^6=5	α^7 = 1	α=2	α^2=4	α^3=3	α^4=6
α^6 **= α^2 + 1 =5**	0	α^6=5	α^7 = 1	α=2	α^2=4	α^3=3	α^4=6	α^5=7

Table 3 below presents the elements of GF(256)

Field Element	Alpha Exponent	Element Polynomial	Exponent Values		Log of Element
			Binary Representation	Decimal Representation	
0	(undefined)	0	00000000	0	(undefined)
1	α^0	α^0	00000001	1	0
2	α^1	α^1	00000010	2	1
3	α^2	α^2	00000100	4	25
4	α^3	α^3	00001000	8	2
5	α^4	α^4	00010000	16	50
6	α^5	α^5	00100000	32	26
7	α^6	α^6	01000000	64	198
8	α^7	α^7	10000000	128	3
9	α^8	$\alpha^4 + \alpha^3 + \alpha^2 + \alpha^0$	00011101	29	223
10	α^9	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha^1$	00111010	58	51
11	α^{10}	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^2$	01110100	116	238
12	α^{11}	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^3$	11101000	232	27
13	α^{12}	$\alpha^7 + \alpha^6 + \alpha^3 + \alpha^2 + \alpha^0$	11001101	205	104
14	α^{13}	$\alpha^7 + \alpha^2 + \alpha^1 + \alpha^0$	10000111	135	199
15	α^{14}	$\alpha^4 + \alpha^1 + \alpha^0$	00010011	19	75
16	α^{15}	$\alpha^5 + \alpha^2 + \alpha^1$	00100110	38	4
17	α^{16}	$\alpha^6 + \alpha^3 + \alpha^2$	01001100	76	100
18	α^{17}	$\alpha^7 + \alpha^4 + \alpha^3$	10011000	152	224
19	α^{18}	$\alpha^5 + \alpha^3 + \alpha^2 + \alpha^0$	00101101	45	14
20	α^{19}	$\alpha^6 + \alpha^4 + \alpha^3 + \alpha^1$	01011010	90	52
21	α^{20}	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^2$	10110100	180	141
22	α^{21}	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha^0$	01110101	117	239
23	α^{22}	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^3 + \alpha^1$	11101010	234	129
24	α^{23}	$\alpha^7 + \alpha^6 + \alpha^3 + \alpha^0$	11001001	201	28
25	α^{24}	$\alpha^7 + \alpha^3 + \alpha^2 + \alpha^1 + \alpha^0$	10001111	143	193
26	α^{25}	$\alpha^1 + \alpha^0$	00000011	3	105
27	α^{26}	$\alpha^2 + \alpha^1$	00000110	6	248

28	α^{27}	$\alpha^3 + \alpha^2$	00001100	12	200
29	α^{28}	$\alpha^4 + \alpha^3$	00011000	24	8
30	α^{29}	$\alpha^5 + \alpha^4$	00110000	48	76
31	α^{30}	$\alpha^6 + \alpha^5$	01100000	96	113
32	α^{31}	$\alpha^7 + \alpha^6$	11000000	192	5
33	α^{32}	$\alpha^7 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha^0$	10011101	157	138
34	α^{33}	$\alpha^5 + \alpha^2 + \alpha^1 + \alpha^0$	00100111	39	101
35	α^{34}	$\alpha^6 + \alpha^3 + \alpha^2 + \alpha^1$	01001110	78	47
36	α^{35}	$\alpha^7 + \alpha^4 + \alpha^3 + \alpha^2$	10011100	156	225
37	α^{36}	$\alpha^5 + \alpha^2 + \alpha^0$	00100101	37	36
38	α^{37}	$\alpha^6 + \alpha^3 + \alpha^1$	01001010	74	15
39	α^{38}	$\alpha^7 + \alpha^4 + \alpha^2$	10010100	148	33
40	α^{39}	$\alpha^5 + \alpha^4 + \alpha^2 + \alpha^0$	00110101	53	53
41	α^{40}	$\alpha^6 + \alpha^5 + \alpha^3 + \alpha^1$	01101010	106	147
42	α^{41}	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^2$	11010100	212	142
43	α^{42}	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha^0$	10110101	181	218
44	α^{43}	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha^1 + \alpha^0$	01110111	119	240
45	α^{44}	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^3 + \alpha^2 + \alpha^1$	11101110	238	18
46	α^{45}	$\alpha^7 + \alpha^6 + \alpha^0$	11000001	193	130
47	α^{46}	$\alpha^7 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha^1 + \alpha^0$	10011111	159	69
48	α^{47}	$\alpha^5 + \alpha^1 + \alpha^0$	00100011	35	29
49	α^{48}	$\alpha^6 + \alpha^2 + \alpha^1$	01000110	70	181
50	α^{49}	$\alpha^7 + \alpha^3 + \alpha^2$	10001100	140	194
51	α^{50}	$\alpha^2 + \alpha^0$	00000101	5	125
52	α^{51}	$\alpha^3 + \alpha^1$	00001010	10	106
53	α^{52}	$\alpha^4 + \alpha^2$	00010100	20	39
54	α^{53}	$\alpha^5 + \alpha^3$	00101000	40	249
55	α^{54}	$\alpha^6 + \alpha^4$	01010000	80	185
56	α^{55}	$\alpha^7 + \alpha^5$	10100000	160	201
57	α^{56}	$\alpha^6 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha^0$	01011101	93	154
58	α^{57}	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^1$	10111010	186	9
59	α^{58}	$\alpha^6 + \alpha^5 + \alpha^3 + \alpha^0$	01101001	105	120
60	α^{59}	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^1$	11010010	210	77
61	α^{60}	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^0$	10111001	185	228

62	α^{61}	$\alpha^6+\alpha^5+\alpha^3+\alpha^2+\alpha^1+\alpha^0$	01101111	111	114
63	α^{62}	$\alpha^7+\alpha^6+\alpha^4+\alpha^3+\alpha^2+\alpha^1$	11011110	222	166
64	α^{63}	$\alpha^7+\alpha^5+\alpha^0$	10100001	161	6
65	α^{64}	$\alpha^6+\alpha^4+\alpha^3+\alpha^2+\alpha^1+\alpha^0$	01011111	95	191
66	α^{65}	$\alpha^7+\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha^1$	10111110	190	139
67	α^{66}	$\alpha^6+\alpha^5+\alpha^0$	01100001	97	98
68	α^{67}	$\alpha^7+\alpha^6+\alpha^1$	11000010	194	102
69	α^{68}	$\alpha^7+\alpha^4+\alpha^3+\alpha^0$	10011001	153	221
70	α^{69}	$\alpha^5+\alpha^3+\alpha^2+\alpha^1+\alpha^0$	00101111	47	48
71	α^{70}	$\alpha^6+\alpha^4+\alpha^3+\alpha^2+\alpha^1$	01011110	94	253
72	α^{71}	$\alpha^7+\alpha^5+\alpha^4+\alpha^3+\alpha^2$	10111100	188	226
73	α^{72}	$\alpha^6+\alpha^5+\alpha^2+\alpha^0$	01100101	101	152
74	α^{73}	$\alpha^7+\alpha^6+\alpha^3+\alpha^1$	11001010	202	37
75	α^{74}	$\alpha^7+\alpha^3+\alpha^0$	10001001	137	179
76	α^{75}	$\alpha^3+\alpha^2+\alpha^1+\alpha^0$	00001111	15	16
77	α^{76}	$\alpha^4+\alpha^3+\alpha^2+\alpha^1$	00011110	30	145
78	α^{77}	$\alpha^5+\alpha^4+\alpha^3+\alpha^2$	00111100	60	34
79	α^{78}	$\alpha^6+\alpha^5+\alpha^4+\alpha^3$	01111000	120	136
80	α^{79}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4$	11110000	240	54
81	α^{80}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha^0$	11111101	253	208
82	α^{81}	$\alpha^7+\alpha^6+\alpha^5+\alpha^2+\alpha^1+\alpha^0$	11100111	231	148
83	α^{82}	$\alpha^7+\alpha^6+\alpha^4+\alpha^1+\alpha^0$	11010011	211	206
84	α^{83}	$\alpha^7+\alpha^5+\alpha^4+\alpha^3+\alpha^1+\alpha^0$	10111011	187	143
85	α^{84}	$\alpha^6+\alpha^5+\alpha^3+\alpha^1+\alpha^0$	01101011	107	150
86	α^{85}	$\alpha^7+\alpha^6+\alpha^4+\alpha^2+\alpha^1$	11010110	214	219
87	α^{86}	$\alpha^7+\alpha^5+\alpha^4+\alpha^0$	10110001	177	189
88	α^{87}	$\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha^1+\alpha^0$	01111111	127	241
89	α^{88}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha^1$	11111110	254	210
90	α^{89}	$\alpha^7+\alpha^6+\alpha^5+\alpha^0$	11100001	225	19
91	α^{90}	$\alpha^7+\alpha^6+\alpha^4+\alpha^3+\alpha^2+\alpha^1+\alpha^0$	11011111	223	92
92	α^{91}	$\alpha^7+\alpha^5+\alpha^1+\alpha^0$	10100011	163	131
93	α^{92}	$\alpha^6+\alpha^4+\alpha^3+\alpha^1+\alpha^0$	01011011	91	56

94	α^{93}	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha^1$	10110110	182	70
95	α^{94}	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^0$	01110001	113	64
96	α^{95}	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^1$	11100010	226	30
97	α^{96}	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^3 + \alpha^0$	11011001	217	66
98	α^{97}	$\alpha^7 + \alpha^5 + \alpha^3 + \alpha^2 + \alpha^1 + \alpha^0$	10101111	175	182
99	α^{98}	$\alpha^6 + \alpha^1 + \alpha^0$	01000011	67	163
100	α^{99}	$\alpha^7 + \alpha^2 + \alpha^1$	10000110	134	195
101	α^{100}	$\alpha^4 + \alpha^0$	00010001	17	72
102	α^{101}	$\alpha^5 + \alpha^1$	00100010	34	126
103	α^{102}	$\alpha^6 + \alpha^2$	01000100	68	110
104	α^{103}	$\alpha^7 + \alpha^3$	10001000	136	107
105	α^{104}	$\alpha^3 + \alpha^2 + \alpha^0$	00001101	13	58
106	α^{105}	$\alpha^4 + \alpha^3 + \alpha^1$	00011010	26	40
107	α^{106}	$\alpha^5 + \alpha^4 + \alpha^2$	00110100	52	84
108	α^{107}	$\alpha^6 + \alpha^5 + \alpha^3$	01101000	104	250
109	α^{108}	$\alpha^7 + \alpha^6 + \alpha^4$	11010000	208	133
110	α^{109}	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha^0$	10111101	189	186
111	α^{110}	$\alpha^6 + \alpha^5 + \alpha^2 + \alpha^1 + \alpha^0$	01100111	103	61
112	α^{111}	$\alpha^7 + \alpha^6 + \alpha^3 + \alpha^2 + \alpha^1$	11001110	206	202
113	α^{112}	$\alpha^7 + \alpha^0$	10000001	129	94
114	α^{113}	$\alpha^4 + \alpha^3 + \alpha^2 + \alpha^1 + \alpha^0$	00011111	31	155
115	α^{114}	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha^1$	00111110	62	159
116	α^{115}	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2$	01111100	124	10
117	α^{116}	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3$	11111000	248	21
118	α^{117}	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^3 + \alpha^2 + \alpha^0$	11101101	237	121
119	α^{118}	$\alpha^7 + \alpha^6 + \alpha^2 + \alpha^1 + \alpha^0$	11000111	199	43
120	α^{119}	$\alpha^7 + \alpha^4 + \alpha^1 + \alpha^0$	10010011	147	78
121	α^{120}	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha^1 + \alpha^0$	00111011	59	212
122	α^{121}	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha^1$	01110110	118	229
123	α^{122}	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^3 + \alpha^2$	11101100	236	172
124	α^{123}	$\alpha^7 + \alpha^6 + \alpha^2 + \alpha^0$	11000101	197	115
125	α^{124}	$\alpha^7 + \alpha^4 + \alpha^2 + \alpha^1 + \alpha^0$	10010111	151	243

126	α^{125}	$\alpha^5 + \alpha^4 + \alpha^1 + \alpha^0$	00110011	51	167
127	α^{126}	$\alpha^6 + \alpha^5 + \alpha^2 + \alpha^1$	01100110	102	87
128	α^{127}	$\alpha^7 + \alpha^6 + \alpha^3 + \alpha^2$	11001100	204	7
129	α^{128}	$\alpha^7 + \alpha^2 + \alpha^0$	10000101	133	112
130	α^{129}	$\alpha^4 + \alpha^2 + \alpha^1 + \alpha^0$	00010111	23	192
131	α^{130}	$\alpha^5 + \alpha^3 + \alpha^2 + \alpha^1$	00101110	46	247
132	α^{131}	$\alpha^6 + \alpha^4 + \alpha^3 + \alpha^2$	01011100	92	140
133	α^{132}	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^3$	10111000	184	128
134	α^{133}	$\alpha^6 + \alpha^5 + \alpha^3 + \alpha^2 + \alpha^0$	01101101	109	99
135	α^{134}	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^3 + \alpha^1$	11011010	218	13
136	α^{135}	$\alpha^7 + \alpha^5 + \alpha^3 + \alpha^0$	10101001	169	103
137	α^{136}	$\alpha^6 + \alpha^3 + \alpha^2 + \alpha^1 + \alpha^0$	01001111	79	74
138	α^{137}	$\alpha^7 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha^1$	10011110	158	222
139	α^{138}	$\alpha^5 + \alpha^0$	00100001	33	237
140	α^{139}	$\alpha^6 + \alpha^1$	01000010	66	49
141	α^{140}	$\alpha^7 + \alpha^2$	10000100	132	197
142	α^{141}	$\alpha^4 + \alpha^2 + \alpha^0$	00010101	21	254
143	α^{142}	$\alpha^5 + \alpha^3 + \alpha^1$	00101010	42	24
144	α^{143}	$\alpha^6 + \alpha^4 + \alpha^2$	01010100	84	227
145	α^{144}	$\alpha^7 + \alpha^5 + \alpha^3$	10101000	168	165
146	α^{145}	$\alpha^6 + \alpha^3 + \alpha^2 + \alpha^0$	01001101	77	153
147	α^{146}	$\alpha^7 + \alpha^4 + \alpha^3 + \alpha^1$	10011010	154	119
148	α^{147}	$\alpha^5 + \alpha^3 + \alpha^0$	00101001	41	38
149	α^{148}	$\alpha^6 + \alpha^4 + \alpha^1$	01010010	82	184
150	α^{149}	$\alpha^7 + \alpha^5 + \alpha^2$	10100100	164	180
151	α^{150}	$\alpha^6 + \alpha^4 + \alpha^2 + \alpha^0$	01010101	85	124
152	α^{151}	$\alpha^7 + \alpha^5 + \alpha^3 + \alpha^1$	10101010	170	17
153	α^{152}	$\alpha^6 + \alpha^3 + \alpha^0$	01001001	73	68
154	α^{153}	$\alpha^7 + \alpha^4 + \alpha^1$	10010010	146	146
155	α^{154}	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha^0$	00111001	57	217
156	α^{155}	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^1$	01110010	114	35
157	α^{156}	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^2$	11100100	228	32

158	α^{157}	$\alpha^7+\alpha^6+\alpha^4+\alpha^2+\alpha^0$	11010101	213	137
159	α^{158}	$\alpha^7+\alpha^5+\alpha^4+\alpha^2+\alpha^1+\alpha^0$	10110111	183	46
160	α^{159}	$\alpha^6+\alpha^5+\alpha^4+\alpha^1+\alpha^0$	01110011	115	55
161	α^{160}	$\alpha^7+\alpha^6+\alpha^5+\alpha^2+\alpha^1$	11100110	230	63
162	α^{161}	$\alpha^7+\alpha^6+\alpha^4+\alpha^0$	11010001	209	209
163	α^{162}	$\alpha^7+\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha^1+\alpha^0$	10111111	191	91
164	α^{163}	$\alpha^6+\alpha^5+\alpha^1+\alpha^0$	01100011	99	149
165	α^{164}	$\alpha^7+\alpha^6+\alpha^2+\alpha^1$	11000110	198	188
166	α^{165}	$\alpha^7+\alpha^4+\alpha^0$	10010001	145	207
167	α^{166}	$\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha^1+\alpha^0$	00111111	63	205
168	α^{167}	$\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha^1$	01111110	126	144
169	α^{168}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^2$	11111100	252	135
170	α^{169}	$\alpha^7+\alpha^6+\alpha^5+\alpha^2+\alpha^0$	11100101	229	151
171	α^{170}	$\alpha^7+\alpha^6+\alpha^4+\alpha^2+\alpha^1+\alpha^0$	11010111	215	178
172	α^{171}	$\alpha^7+\alpha^5+\alpha^4+\alpha^1+\alpha^0$	10110011	179	220
173	α^{172}	$\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^1+\alpha^0$	01111011	123	252
174	α^{173}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^2+\alpha^1$	11110110	246	190
175	α^{174}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^0$	11110001	241	97
176	α^{175}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha^1+\alpha^0$	11111111	255	242
177	α^{176}	$\alpha^7+\alpha^6+\alpha^5+\alpha^1+\alpha^0$	11100011	227	86
178	α^{177}	$\alpha^7+\alpha^6+\alpha^4+\alpha^3+\alpha^1+\alpha^0$	11011011	219	211
179	α^{178}	$\alpha^7+\alpha^5+\alpha^3+\alpha^1+\alpha^0$	10101011	171	171
180	α^{179}	$\alpha^6+\alpha^3+\alpha^1+\alpha^0$	01001011	75	20
181	α^{180}	$\alpha^7+\alpha^4+\alpha^2+\alpha^1$	10010110	150	42
182	α^{181}	$\alpha^5+\alpha^4+\alpha^0$	00110001	49	93
183	α^{182}	$\alpha^6+\alpha^5+\alpha^1$	01100010	98	158
184	α^{183}	$\alpha^7+\alpha^6+\alpha^2$	11000100	196	132
185	α^{184}	$\alpha^7+\alpha^4+\alpha^2+\alpha^0$	10010101	149	60
186	α^{185}	$\alpha^5+\alpha^4+\alpha^2+\alpha^1+\alpha^0$	00110111	55	57
187	α^{186}	$\alpha^6+\alpha^5+\alpha^3+\alpha^2+\alpha^1$	01101110	110	83
188	α^{187}	$\alpha^7+\alpha^6+\alpha^4+\alpha^3+\alpha^2$	11011100	220	71
189	α^{188}	$\alpha^7+\alpha^5+\alpha^2+\alpha^0$	10100101	165	109

190	α^{189}	$\alpha^6 + \alpha^4 + \alpha^2 + \alpha^1 + \alpha^0$	01010111	87	65
191	α^{190}	$\alpha^7 + \alpha^5 + \alpha^3 + \alpha^2 + \alpha^1$	10101110	174	162
192	α^{191}	$\alpha^6 + \alpha^0$	01000001	65	31
193	α^{192}	$\alpha^7 + \alpha^1$	10000010	130	45
194	α^{193}	$\alpha^4 + \alpha^3 + \alpha^0$	00011001	25	67
195	α^{194}	$\alpha^5 + \alpha^4 + \alpha^1$	00110010	50	216
196	α^{195}	$\alpha^6 + \alpha^5 + \alpha^2$	01100100	100	183
197	α^{196}	$\alpha^7 + \alpha^6 + \alpha^3$	11001000	200	123
198	α^{197}	$\alpha^7 + \alpha^3 + \alpha^2 + \alpha^0$	10001101	141	164
199	α^{198}	$\alpha^2 + \alpha^1 + \alpha^0$	00000111	7	118
200	α^{199}	$\alpha^3 + \alpha^2 + \alpha^1$	00001110	14	196
201	α^{200}	$\alpha^4 + \alpha^3 + \alpha^2$	00011100	28	23
202	α^{201}	$\alpha^5 + \alpha^4 + \alpha^3$	00111000	56	73
203	α^{202}	$\alpha^6 + \alpha^5 + \alpha^4$	01110000	112	236
204	α^{203}	$\alpha^7 + \alpha^6 + \alpha^5$	11100000	224	127
205	α^{204}	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha^0$	11011101	221	12
206	α^{205}	$\alpha^7 + \alpha^5 + \alpha^2 + \alpha^1 + \alpha^0$	10100111	167	111
207	α^{206}	$\alpha^6 + \alpha^4 + \alpha^1 + \alpha^0$	01010011	83	246
208	α^{207}	$\alpha^7 + \alpha^5 + \alpha^2 + \alpha^1$	10100110	166	108
209	α^{208}	$\alpha^6 + \alpha^4 + \alpha^0$	01010001	81	161
210	α^{209}	$\alpha^7 + \alpha^5 + \alpha^1$	10100010	162	59
211	α^{210}	$\alpha^6 + \alpha^4 + \alpha^3 + \alpha^0$	01011001	89	82
212	α^{211}	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^1$	10110010	178	41
213	α^{212}	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^0$	01111001	121	157
214	α^{213}	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^1$	11110010	242	85
215	α^{214}	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^0$	11111001	249	170
216	α^{215}	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^3 + \alpha^2 + \alpha^1 + \alpha^0$	11101111	239	251
217	α^{216}	$\alpha^7 + \alpha^6 + \alpha^1 + \alpha^0$	11000011	195	96
218	α^{217}	$\alpha^7 + \alpha^4 + \alpha^3 + \alpha^1 + \alpha^0$	10011011	155	134
219	α^{218}	$\alpha^5 + \alpha^3 + \alpha^1 + \alpha^0$	00101011	43	177
220	α^{219}	$\alpha^6 + \alpha^4 + \alpha^2 + \alpha^1$	01010110	86	187

221	α^{220}	$\alpha^7+\alpha^5+\alpha^3+\alpha^2$	10101100	172	204
222	α^{221}	$\alpha^6+\alpha^2+\alpha^0$	01000101	69	62
223	α^{222}	$\alpha^7+\alpha^3+\alpha^1$	10001010	138	90
224	α^{223}	$\alpha^3+\alpha^0$	00001001	9	203
225	α^{224}	$\alpha^4+\alpha^1$	00010010	18	89
226	α^{225}	$\alpha^5+\alpha^2$	00100100	36	95
227	α^{226}	$\alpha^6+\alpha^3$	01001000	72	176
228	α^{227}	$\alpha^7+\alpha^4$	10010000	144	156
229	α^{228}	$\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha^0$	00111101	61	169
230	α^{229}	$\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^1$	01111010	122	160
231	α^{230}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^2$	11110100	244	81
232	α^{231}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^2+\alpha^0$	11110101	245	11
233	α^{232}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^2+\alpha^1+\alpha^0$	11110111	247	245
234	α^{233}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^1+\alpha^0$	11110011	243	22
235	α^{234}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^1+\alpha^0$	11111011	251	235
236	α^{235}	$\alpha^7+\alpha^6+\alpha^5+\alpha^3+\alpha^1+\alpha^0$	11101011	235	122
237	α^{236}	$\alpha^7+\alpha^6+\alpha^3+\alpha^1+\alpha^0$	11001011	203	117
238	α^{237}	$\alpha^7+\alpha^3+\alpha^1+\alpha^0$	10001011	139	44
239	α^{238}	$\alpha^3+\alpha^1+\alpha^0$	00001011	11	215
240	α^{239}	$\alpha^4+\alpha^2+\alpha^1$	00010110	22	79
241	α^{240}	$\alpha^5+\alpha^3+\alpha^2$	00101100	44	174
242	α^{241}	$\alpha^6+\alpha^4+\alpha^3$	01011000	88	213
243	α^{242}	$\alpha^7+\alpha^5+\alpha^4$	10110000	176	233
244	α^{243}	$\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha^0$	01111101	125	230
245	α^{244}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^1$	11111010	250	231
246	α^{245}	$\alpha^7+\alpha^6+\alpha^5+\alpha^3+\alpha^0$	11101001	233	173
247	α^{246}	$\alpha^7+\alpha^6+\alpha^3+\alpha^2+\alpha^1+\alpha^0$	11001111	207	232
248	α^{247}	$\alpha^7+\alpha^1+\alpha^0$	10000011	131	116
249	α^{248}	$\alpha^4+\alpha^3+\alpha^1+\alpha^0$	00011011	27	214

250	α^{249}	$\alpha^5 + \alpha^4 + \alpha^2 + \alpha^1$	00110110	54	244
251	α^{250}	$\alpha^6 + \alpha^5 + \alpha^3 + \alpha^2$	01101100	108	234
252	α^{251}	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^3$	11011000	216	168
253	α^{252}	$\alpha^7 + \alpha^5 + \alpha^3 + \alpha^2 + \alpha^0$	10101101	173	80
254	α^{253}	$\alpha^6 + \alpha^2 + \alpha^1 + \alpha^0$	01000111	71	88
255	α^{254}	$\alpha^7 + \alpha^3 + \alpha^2 + \alpha^1$	10001110	142	175
	α^{255}	α^0	00000001	1	