

Computational Design of BCH-Codes and Their Applications in the Data Security



Muhammad Asif

**Department of Mathematics
Quaid-i-Azam University
Islamabad, Pakistan**

2020

Computational Design of BCH-Codes and Their Applications in the Data Security



By

Muhammad Asif

Supervised

By

Prof. Dr. Tariq Shah

Department of Mathematics

Quaid-i-Azam University

Islamabad, Pakistan

2020

Computational Design of BCH-Codes and Their Applications in the Data Security



A Thesis Submitted to the Department of Mathematics,
Quaid-i-Azam University, Islamabad, in the partial fulfillment of
the requirement for the degree of

Doctor of Philosophy

in

Mathematics

By

Muhammad Asif

Department of Mathematics

Quaid-i-Azam University

Islamabad, Pakistan

2020

Author's Declaration

I, Muhammad Asif, hereby state that my PhD thesis titled Computational Design of BCH-Codes and Their Applications in the Data Security is my own work and has not been submitted previously by me for taking any degree from the Quaid-I-Azam University Islamabad, Pakistan or anywhere else in the country/world.

At any time if my statement is found to be incorrect even after my graduate the university has the right to withdraw my PhD degree.



Name of Student: Muhammad Asif

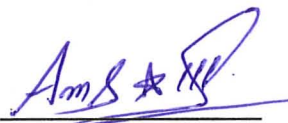
Date: 08-09-2020

Plagiarism Undertaking

I solemnly declare that research work presented in the thesis titled **“Computational Design of BCH-Codes and Their Applications in the Data Security”** is solely my research work with no significant contribution from any other person. Small contribution/help wherever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero-tolerance policy of the HEC and **Quaid-i-Azam University** towards plagiarism. Therefore, I as an Author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred/cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even afterward of PhD degree, the University reserves the rights to withdraw/revoke my PhD degree and that HEC and the University has the right to publish my name on the HEC/University Website on which names of students are placed who submitted plagiarized thesis.

Student/Author Signature: 

Name: **Muhammad Asif**

Computational Design of BCH-Codes and Their Applications in the Data Security

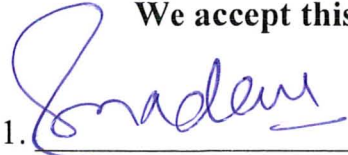
By

Muhammad Asif

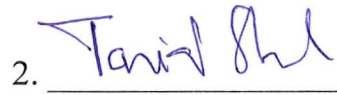
CERTIFICATE

A THESIS SUBMITTED IN THE PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF THE
DOCTOR OF PHILOSOPHY IN MATHEMATICS

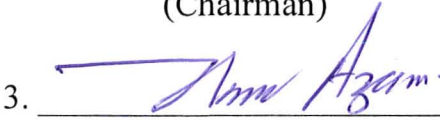
We accept this thesis as conforming to the required standard

1. 

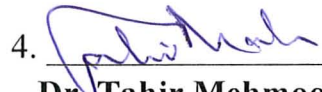
Prof. Dr. Sohail Nadeem
(Chairman)

2. 

Prof. Dr. Tariq Shah
(Supervisor)

3. 

Prof. Dr. Akbar Azam
(External Examiner)

4. 

Dr. Tahir Mehmood
(External Examiner)

Department of Mathematics, COMSATS
University, Park Road, Chak Shahzad,
Islamabad.

Department of Mathematics & Statistics,
Faculty of Basics Applied Sciences,
International Islamic University, Islamabad.

Department of Mathematics
Quaid-I-Azam University
Islamabad, Pakistan
2020

Certificate of Approval

This is to certify that the research work presented in this thesis entitled Computational Design of BCH-Codes and Their Applications in the Data Security was conducted by Mr. Muhammad Asif under the kind supervision of Dr. Tariq Shah. No part of this thesis has been submitted anywhere else for any other degree. This thesis is submitted to the Department of Mathematics, Quaid-i-Azam University, Islamabad in partial fulfillment of the requirements for the degree of Doctor of Philosophy in field of Mathematics from Department of Mathematics, Quaid-i-Azam University Islamabad, Pakistan.

Student Name: Muhammad Asif

Signature: 

External committee:

a) External Examiner 1:

Name: **Dr. Akbar Azam**

Designation: Professor

Office Address: Department of Mathematics, COMSATS University, Park Road, Chak Shahzad, Islamabad.

Signature: 

b) External Examiner 2:

Name: **Dr. Tahir Mehmood**

Designation: Assistant Professor

Office Address: Department of Mathematics & Statistics, Faculty of Basics Applied Sciences, International Islamic University, Islamabad.

Signature: 

c) Internal Examiner

Name: **Dr. Tariq Shah**

Designation: Professor

Office Address: Department of Mathematics, QAU Islamabad.

Signature: 

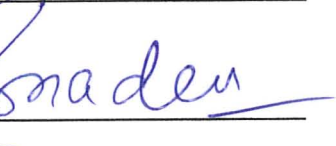
Supervisor Name:

Prof. Dr. Tariq Shah

Signature: 

Name of Dean/ HOD

Prof. Dr. Sohail Nadeem

Signature: 

DEDICATED

TO

MY

BELOVED

PARENTS

Acknowledgement

All praise for Almighty **Allah**, the creator and the Merciful Lord, who guides me in darkness, helps me in difficulties and enables me to reach the ultimate stage with courage. All of my veneration and devotion goes to our beloved **Prophet Hazrat Muhammad (S.A.W)** the source of humanity, kindness and guidance for the whole creatures and who declared it an obligatory duty of every Muslim to seek acquire knowledge.

I express deepest gratitude to my respected supervisor **Prof. Dr. Tariq Shah** for his intellectual guidance, constant encouragement, valuable suggestions, and inexhaustible inspiration throughout my research work. He was the backbone of this research work with constructive criticism and extensive discussions. In short, his tireless work, unique way of research and devotion to his profession cannot be expressed in words.

I wish to express my heartiest thanks and gratitude to my parents, my father **Sep Muhammad Rafique Shaheed** who sacrifice himself for this country and got “**SHAHADAT**” during service in Pak Army. My mother, the ones who can never ever be thanked enough for the overwhelming love, kindness, and care they bestow upon me. She gave me love of father too, and support me in every difficulty, without her proper guidance it would not been possible for me to complete my higher education. I am also thankful to my younger brothers **Muhammad Rashid** and **Muhammad Nasir** for their love and care in every moment of my PhD.

I would like to express my gratitude to all the respected teachers. Sir Tayyab, Sir Tasawar, Sir Khalid, Sir Safdar, Sir Muzaffar, Sir Aslam, they are all those people who made me what I am today, they polished me at different stages of my life and taught me whatever I am today.

I gratefully acknowledge my seniors, my friends (Dr. Yasir Naseer, Mr. Mubasher Umer, Muhammad Tanveer, Dr Kashif Shafiq, Dr. Sajid, Dr. Zafar Saeed, Dawood Shah) and my PhD fellows for their brilliant ideas and important contribution in refining my research work. Their professional guidance has nourished and polished my intellectual skills and I will always remain thankful to them.

I am also thankful to administrative staff of mathematics department and all faculty for their support at every time.

In the end, I would like to all my research fellows and to those people who directly and indirectly helped me during my research work.

Muhammad Asif

08-09-2020

Preface

Due to innovations in communication technologies, digital medium is extensively used across the World. Large amount of information in digital form is stockpiled in digital libraries. The error free transportation of digital data through untrustworthy channels is a great challenge. Accordingly, in information theory, telecommunication, computer science and algebraic coding theory, an error correction code or error correcting code is used for controlling errors in data over unreliable or noisy communication channels. Because of error correcting codes, the communication is made over the short and long distances without any obstacle. Thus, it made possible the gigabit data transmission over the wireless communication mediums. Indeed, it is the fundamental part of the modern communication systems and essentially utilized in hardware level implementations of intelligent and smart machines like telecom equipment, highly sensitive video cameras, optical devices, and scanners.

Development of data transferring codes were started with the first article [31] of Claude Shannon in 1948. He explained that, every communication channel has some capacity. If the rate of data transmission is smaller than capacity, then design of communication system for the channel is possible with the help of data transmission codes. This system has least probability of output errors, but Shannon did not give the method for the construction of such type of codes. In 1950, for this purpose Hamming [14] and Golay [9] introduced cyclic block codes known as binary hamming and Golay codes respectively. These classes of codes have the capability to detect up to two errors and correct one error. Furthermore, these codes have fascinating features and can be easily encoded and decoded but are not suitable for multiple errors. In 1953, Muller [18] introduced a multiple error correcting codes technique and Reed [26] developed decoding technique of such type of codes. Yet, Shannon's hypothesis remained unresolved.

Cyclic codes are one of the dynamic class of error correcting codes. In 1957, Prange [33] initiated an idea of cyclic codes in two symbols. In addition, Prange [24] used the coset equivalence for decoding the group codes in 1959. After that, a big development in the theory of cyclic codes was made to correct burst along with random errors initiated by various researchers. The cyclic codes were initially developed over binary field \mathbb{Z}_2 and into its Galois field extension $GF(2^m)$. Though, it was further extended over the prime field \mathbb{Z}_p and into its Galois field extension $GF(p^m)$. The remarkable development in coding theory began when Hocquenghem [10], Bose and Chaudhuri [3] explained the large class of codes which correct multiple errors known as BCH codes in 1960. They explained the BCH codes over Galois field. These codes are generalization of binary Hamming codes. The advantage of BCH code is that,

Fundamentally the BCH codes are utilized for only data transmission, but not for data security. In this study we have given the idea that, BCH codes can be used for data security. Accordingly, by BCH codes over Galois field and Galois ring a couple of techniques are devised to modify AES algorithm. Accordingly, this modified AES algorithm tested on text and image data, the results assured the appropriate level of security.

This thesis consists of seven chapters.

In Chapter one, some important notions of algebraic structures and error correcting codes are explained which are necessary for understanding further chapters.

In Chapter two, initially we have given details on obtaining the maximal cyclic subgroup of group of units of a Galois ring through computational method. Afterward the new computational encoding scheme of BCH code over Galois ring is introduced. This novel computational approach of encoding of BCH codes provides generator polynomial for any length n corresponding to each designed distance d . Furthermore, the encoding of BCH codes over Galois field has also been explained with the help of reduction map. Another outcome of this study is that one can find the dimensions of primitive BCH codes for any length and designed distance.

In Chapter three, using C# computer language a computational decoding scheme for BCH codes over Galois ring has been designed by which Berlekamp Massey decoding algorithm of BCH codes over Galois field is employed to correct the errors. Indeed, this modified Berlekamp Massey decoding algorithm is designed for large length BCH codes over Galois field. The special feature of this study is the syndrome calculation with computational approach. Thus, decoding of BCH and RS codes over Galois ring by using modified Berlekamp Massey algorithm has been ensured.

In Chapter four, BCH codes have been utilized to improve the AES algorithm. BCH codes have been utilized as a secret key in round key addition step of AES algorithm. In addition, using BCH codes, the maximum distance separable matrix has been constructed and applied in mixed column matrix step in modified AES algorithm. Thus, this modified AES algorithm has been applied in image encryption and different analyses on encrypted image have been performed. The comparison of results of encrypted image by using original and modified AES algorithms have been discussed.

In Chapter five, The AES algorithm is modified. Initially we use BCH codes and calculated secret keys for each round in AES algorithm. In second step, mixed column matrices have been computed by using BCH codes for each round. This modified AES algorithm has been used for text encryption and then applied avalanche effect to cipher text. NIST statistical test have been applied on proposed text encryption scheme.

Contents

1 Algebraic Notions and Error Correcting Codes	4
1.1 Finite Rings and Finite Fields	4
1.1.1 Polynomial Ring	5
1.1.2 Galois Field	5
1.1.3 Galois Ring	6
1.1.4 Linear Spaces	7
1.2 Fundamentals of Error Correcting Codes	7
1.2.1 Codes	7
1.2.2 Linear Codes	9
1.2.3 Hamming Codes	11
1.2.4 Cyclic Codes	12
1.2.5 BCH Codes	13
2 BCH-Codes over Galois Ring and Galois Field: Computational Encoding Approach	15
2.1 BCH Codes over Galois Ring: Encoding	16
2.1.1 Maximal Cyclic Subgroup over Galois Ring	17
2.1.2 Maximal Cyclic Subgroup with Computational Approach	19
2.1.3 Generator Polynomial of BCH Codes using Maximal Cyclic Subgroup	20
2.2 Computationally Encoding of BCH Codes over Galois Ring	35
2.2.1 Algorithm for Designing BCH Codes over Galois Ring	36
2.2.2 Explanation of Algorithm 1 with Example	44

4.4.1	Contrast	82
4.4.2	Correlation	82
4.4.3	Energy	83
4.4.4	Homogeneity	83
4.4.5	Entropy	83
4.4.6	Histogram Analyses	85
5	Symmetric Block Cipher and BCH Codes: A Text Encryption Application	87
5.1	Construction of MDS Matrices using BCH Codes	88
5.2	Construction of Secret Keys using BCH Codes	91
5.3	Application of BCH Codes in Text Data	95
5.4	Text Encryption Analyses	97
5.4.1	Avalanche Effect	97
5.4.2	NIST Statistical Test	100
5.4.3	Ciphertext Attack	101
5.4.4	Known Plaintext Attack	101
6	A Nonlinear Component Design in Symmetric Block Cipher with Image Encryption Application	102
6.1	Construction of Nonlinear Component using Galois Ring and Galois Field	102
6.1.1	Scheme for Nonlinear Component over the Galois Ring	103
6.1.2	Scheme for Nonlinear Component over the Galois Field	108
6.2	Application of Nonlinear Component in Image Encryption	114
6.2.1	Encryption Procedure Explanation	127
6.2.2	Statistical Analysis	129
6.2.3	Histogram Analysis	130
6.2.4	Analyses Discussion	132
7	Conclusion	133
8	References	135
9	Appendix	139



Chapter 1

Algebraic Notions and Error Correcting Codes

In this chapter, we present some basic notions necessary for the understanding of thesis and have two phases. In the first phase, we discuss some algebraic concepts of rings and fields. In the second phase, we present some fundamentals of error-correcting codes.

1.1 Finite Rings and Finite Fields

Let a non empty set \mathcal{R} with two operations addition '+' and multiplication '.' is called **ring** if it satisfies the following conditions.

- i) \mathcal{R} is the abelian group under addition '+'.
- ii) \mathcal{R} is semigroup under multiplication '.'.
- iii) Multiplication '.' is distributive with respect to addition in \mathcal{R} .

If $r_1, r_2 \in \mathcal{R}$, and $r_1 \cdot r_2 = r_2 \cdot r_1$, then it is called a **commutative ring**. Let \mathcal{R} be a ring, and a non zero element r_1 of \mathcal{R} is said to be **zero divisor** from left if there exists a non zero element r_2 of \mathcal{R} such that $r_1 r_2 = 0$ where r_2 is right zero divisor. If \mathcal{R} is commutative ring, then r_1 and r_2 are zero-divisors of each other. A ring \mathcal{R} is called an **integral domain** if it does not contain any zero divisor. Let \mathcal{R} be a commutative ring with identity, and a non zero element $r_1 \in \mathcal{R}$ is a **unit element** if there exists an element $r_2 \in \mathcal{R}$ such that $r_2 r_1 = r_1 r_2 = 1$. We denote a set of units elements by $U(\mathcal{R})$. Let r be an element of ring \mathcal{R} , then it is said to be **nilpotent** if

The order of $GF(p^m)$ is p^m . The Galois field $GF(p^m)$ is denoted by \mathcal{K} , and \mathcal{K}^* denotes the group of non zero elements of Galois field. If $q = p^n$ where p is prime and $n \in \mathbb{Z}^+$, $f(x)$ is primitive irreducible polynomial of degree m then,

$$\frac{F_q[X]}{(f(x))} \simeq F_{q^m} = GF(q^m).$$

Example 1 : If $q = 2$ and $m = 3$ then,

$$\frac{\mathbb{Z}_2[X]}{\langle x^3 + x + 1 \rangle} \simeq GF(2^3) = GF(8), \quad (1.5)$$

where $x^3 + x + 1$ is a primitive irreducible polynomial with primitive root α in $\mathbb{Z}_2[X]$,

$$GF(8) = \{r + s\alpha + t\alpha^2 : r, s, t \in \mathbb{Z}_2, 1 + \alpha + \alpha^3 = 0\}.$$

Therefore, elements of $GF(8)$ are shown in Table 1.1,

0	α	α^2	$\alpha^3 = \alpha + 1$	$\alpha^4 = \alpha + \alpha^2$	$\alpha^5 = 1 + \alpha + \alpha^2$	$\alpha^6 = 1 + \alpha^2$	$\alpha^7 = 1$
---	----------	------------	-------------------------	--------------------------------	------------------------------------	---------------------------	----------------

Table 1.1: Elements of Galois field $GF(2^3)$

1.1.3 Galois Ring

Let n, m be any positive integers and p be a prime number, where m be the degree of basic irreducible polynomial $f(x)$ then Galois ring be defined as,

$$\frac{\mathbb{Z}_{p^n}[x]}{(f(x))} = \{p_0 + p_1x + p_2x^2 + \dots + p_{m-1}x^{m-1} : p_0, p_1, \dots, p_{m-1} \in \mathbb{Z}_{p^n}\} \simeq GR(p^n, m). \quad (1.6)$$

It is the Galois extension ring of \mathbb{Z}_{p^n} having p^{nm} elements. Galois ring $GR(p^n, m)$ is denoted by \mathcal{R} , and \mathcal{R}^* denotes the group of units of Galois ring.

are known as **codewords**. Trivial code contains an only a single element. If each element of the code C can be written in the form $vvv\dots v$ for some $v \in V$, then C is called **repetition code**. The q -ary repetition code contains exactly q codewords. Let $w, v \in V^n$, $w = w_1w_2w_3\dots w_n$ and $v = v_1v_2v_3\dots v_n$ then the **Hamming distance** $d(w, v)$ between the vectors w and v is defined as,

$$d(w, v) = |\{i : w_i \neq v_i\}|. \quad (1.7)$$

The least distance between any two different codewords in C is called **minimum distance**, and it is denoted by $d(C)$,

$$d(C) = \min\{d(w, v) : \text{for all } w, v \in C \text{ and } w \neq v\}. \quad (1.8)$$

For example, the minimum distance of $C = \{000, 101, 100, 111\}$ is 1.

Theorem 2 [19]: *Suppose that code C having minimum distance $d(C)$. Let $t = \lfloor \frac{d-1}{2} \rfloor$ then the errors detected in the received word are $d-1$, and the errors can be corrected in any received word are t .*

Every code C is denoted by $(n, M, d(C))$, where n is the length of code C and M indicates the number of codewords in C and $d(C)$ represents the minimum distance of C . The code C is called **good code** if it satisfies following conditions:

- i) The length n of the code is smaller.
- ii) The size of M is very large.
- iii) $d(C)$ of the code is also large.

The length of the code smaller means that transmission of code is very fast, and the cost of the code is very low. The large M means that we can send more variety of messages, and $d(C)$ of the code is large, which implies that we can correct greater number of errors. The main task of the algebraic theorists is to find those codes whose size M and the minimum distance is maximum for fixed-length n . Suppose $C \subset V^n$ is a code having minimum distance $2t + 1$. If for each $w \in V^n$, there exist $v \in V^n$ such that $d(w, v) \leq t$, then C is known as **perfect code**. For example, binary code $\{000, 111\}$ is a perfect code with minimum distance 3.

perpendicular to itself and perpendicular to each element of C , then a code C is called **self orthogonal**, or if $C^\perp \supseteq C$, then it is also self orthogonal. Let C be the $[n, k]$ code, \mathcal{H} is the generator matrix having order $(n - k) \times n$ of C^\perp . The generator matrix of C^\perp is also known as the **parity check matrix** of C .

Theorem 4 [19]: *If C is a $[n, k]$ code over the field \mathcal{F} and \mathcal{H} is generator matrix of C^\perp then,*

$$C = \{w \in \mathcal{F}^n : w\mathcal{H}^T = 0 = \mathcal{H}w^T\}. \quad (1.12)$$

Theorem 5 [19]: *If C is a $[n, k]$ code, \mathcal{G} is a generator matrix, and \mathcal{H} is a parity check matrix of the code C then,*

$$\mathcal{G}\mathcal{H}^T = 0 = \mathcal{H}\mathcal{G}^T. \quad (1.13)$$

Conversely, suppose that \mathcal{G} is a $k \times n$ matrix, and \mathcal{H} is $(n - k) \times n$ matrix such that $\mathcal{G}\mathcal{H}^T = 0$. The \mathcal{H} is a parity check matrix of the code C if and only if \mathcal{G} is the generator matrix of C , where the rank of \mathcal{G} is k , and rank of the \mathcal{H} matrix is $n - k$.

Example 6 : *If the code $C = \{000, 111\}$ then $C^\perp = \{110, 000, 101, 011\}$. The generator matrix of C is,*

$$\mathcal{G} = [1 \ 1 \ 1], \quad (1.14)$$

and parity check matrix of C is,

$$\mathcal{H} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}. \quad (1.15)$$

Theorem 7 [19]: *Suppose C is $[n, k]$ code. If C has generator matrix as,*

$$\mathcal{G} = [I_k \ :B], \quad (1.16)$$

where B is $k \times (n - k)$ matrix, then the parity check \mathcal{H} is defined as

$$\mathcal{H} = [-B^T \ : I_{n-k}]. \quad (1.17)$$

If parity check matrix \mathcal{H} is defined as,

$$\mathcal{H} = [A \ : I_{n-k}], \quad (1.18)$$

Therefore, $\text{Ham}(2, 2)$ is $\{111, 000\}$ repetition code.

Theorem 9 [19]: *Ham(m, q) is a perfect code with minimum distance 3.*

1.2.4 Cyclic Codes

In error-correcting codes, properties of cyclic codes are very interesting than general linear codes. There is a huge class of necessary codes that are related to cyclic codes. The mapping $\sigma: \mathcal{F}^n \rightarrow \mathcal{F}^n$ is defined by

$$\sigma(u_1, u_2, u_3, \dots, u_n) = (u_n, u_1, u_2, \dots, u_{n-1}), \quad (1.23)$$

is called a **cyclic shift**. If C is the subset of \mathcal{F}^n and for each element u of C which implies $\sigma(u) \in C$ then it is called **cyclic code**. For example, the code $C = \{100, 010, 001, 000\}$ is cyclic code over \mathcal{F}_2 . Suppose that

$$\mathcal{F}[X]_n = \{\alpha_0 + \alpha_1x + \alpha_2x^2 + \dots + \alpha_{n-1}x^{n-1} : \alpha_i \in \mathcal{F}\}, \quad (1.24)$$

is the set consisting of all polynomials having degree smaller than n over \mathcal{F} .

The mapping $\rho: \mathcal{F}^n \rightarrow \mathcal{F}[X]_n$ is defined as,

$$\rho(u) = u(x) \text{ for all } u = (u_0, u_1, u_2, \dots, u_{n-1}) \in \mathcal{F}^n, \quad (1.25)$$

and this is an isomorphism. Let $\mathcal{F}[X]$ be the polynomial ring over \mathcal{F} and $h(x) \in \mathcal{F}[X]$ be an irreducible element over the field \mathcal{F} , the quotient ring,

$$\frac{\mathcal{F}[X]}{\langle h(x) \rangle} = \{\alpha_0 + \alpha_1t + \alpha_2t^2 + \dots + \alpha_{n-1}t^{n-1} : \alpha_i \in \mathcal{F}\}, \quad (1.26)$$

is a field, such that $t = x + \langle h(x) \rangle$, $h(t) = 0$ and $h(x)$ be the n degree polynomial. Let $h(x) = x^n - 1$, then the quotient ring is,

$$\frac{\mathcal{F}[X]}{\langle x^n - 1 \rangle} = \{\alpha_0 + \alpha_1t + \alpha_2t^2 + \dots + \alpha_{n-1}t^{n-1} : \alpha_0, \alpha_1, \alpha_2, \dots, \alpha_{n-1} \in \mathcal{F}\}, \quad (1.27)$$

equivalently C is the null space of \mathcal{H} ,

$$\mathcal{H} = \begin{bmatrix} 1 & \xi^c & \xi^{2c} & \dots & \xi^{(n-1)c} \\ 1 & \xi^{c+1} & \xi^{2(c+1)} & \dots & \xi^{(n-1)(c+1)} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & \xi^{c+d-2} & \xi^{2(c+d-2)} & \dots & \xi^{(n-1)(c+d-2)} \end{bmatrix}, \quad (1.29)$$

where \mathcal{H} is $(d-1) \times n$ quasi parity check matrix over \mathcal{F}_{q^m} .

Theorem 13 [19]: Let C be a BCH code with designed distance d then the minimum distance $d(C)$ is greater than or equal to designed distance, $d(C) \geq d$.

Chapter 2

BCH-Codes over Galois Ring and Galois Field: Computational Encoding Approach

In this chapter, the encoding of primitive BCH code over the Galois ring and Galois field with the computational approach are explained. The novel approach is introduced to overcome the problems in the construction of generator polynomial and determination the dimension of the BCH code over the Galois ring and Galois field. A modern technique is developed in such a way that the data can be encoded during transmission over the Galois field or Galois ring. The selection of schemes is based on a better code rate and improved error correction capability of the chosen code. Our computational method provides the luxury of sorting this problem computationally to construct codes and dimension over Galois ring. This chapter consists of three sections, initially, an introduction of encoding of BCH codes over Galois ring and construction of maximal cyclic subgroups with the computational approach is determined. In the second part, an algorithm for computing the generator polynomial of BCH codes over the Galois ring is explained. In the last part, the encoding of BCH codes over the Galois field and dimension of primitive BCH codes with the help of computer language is calculated.

Theorem 17 [15]: Let $R_p(\xi) = \bar{\xi}$ generates cyclic subgroup of order n in group of unit elements of $GF(p^r)$. Then ξ generate the cyclic subgroup of \mathcal{R}^* of order $n.d$ for $d \geq 1$ and the maximal cyclic subgroup G_n is generated by ξ^d .

Lemma 18 [15]: Let ξ be the primitive element of G_n . Then the differences $\xi^{m_1} - \xi^{m_2}$ are unit elements in the ring \mathcal{R} if $0 \leq m_1 \neq m_2 \leq n - 1$.

Theorem 19 [15]: The minimum distance of the BCH code is greater than or equal to $2t + 1$.

Remark 20 : If α generates the elements of Galois field $GF(p^m)$, then $\alpha^{p^{n-1}}$ generates the elements of the maximal cyclic subgroup in corresponding Galois ring $GR(p^n, m)$.

2.1.1 Maximal Cyclic Subgroup over Galois Ring

The BCH codes over the Galois ring are calculated corresponding to each Galois field. For Galois fields $GF(p^m)$ of order p^m , there are many Galois rings $GR(p^n, m)$ of order p^{mn} , where $n \in \mathbb{Z}^+$ and m is the degree of monic irreducible polynomial $f(x)$. The polynomial $f(x)$ is an irreducible in the Galois ring, and $\overline{f(x)} = R_p(f(x))$ is primitive irreducible polynomial in the corresponding Galois field. The maximal cyclic subgroup is calculated by the following steps:

Step 1: Select the monic irreducible polynomial over \mathbb{Z}_n , where $n = p^m$.

Step 2: Find the order of root of an irreducible polynomial over Galois ring.

Step 3: Divide the order of root of an irreducible polynomial by order of the maximal cyclic subgroup.

Step 4: Take the output of Step 3 as a power of root of an irreducible polynomial.

Step 5: Select the output of Step 4 as a generator of the maximal cyclic subgroup.

Step 6: Construct all the elements of the maximal cyclic subgroup from a generator of the maximal cyclic subgroup.

Example 21 : Choose $n = 7$ in G_n over \mathbb{Z}_8 , then consider Galois ring $GR(8, 3) \simeq \frac{\mathbb{Z}_8[x]}{\langle x^3+x+1 \rangle}$, where $f(x) = x^3 + x + 1$ is monic irreducible polynomial over \mathbb{Z}_8 . Let α be the root of the polynomial $f(x)$, then $f(\alpha) = 0$. Now find the order of α in Galois ring $GR(8, 3)$, which implies that $\alpha^3 = -\alpha - 1 = 7\alpha + 7$, remaining powers of α are shown as,

degree 5 over \mathbb{Z}_8 , $f(x) = x^5 + 5x^2 + 5$. Therefore, $G_{31} = \langle \beta = \alpha^4 \rangle$.

$\beta = \alpha^4$	$\beta^2 = 3\alpha^3 + \alpha^2 + 1$	$\beta^3 = 2\alpha^4 + 3\alpha^3 + \alpha^2 + 3\alpha$
$\beta^4 = \alpha^4 + \alpha^3 + 4\alpha^2 + 3\alpha + 3$	$\beta^5 = 6\alpha^4 + 7\alpha^3 + 5\alpha^2 + 4\alpha + 2$	$\beta^6 = 7\alpha^4 + \alpha^3 + 7\alpha^2 + 7\alpha + 2$
$\beta^7 = 5\alpha^4 + 2\alpha^3 + 7\alpha^2 + 5\alpha + 4$	$\beta^8 = 2\alpha^4 + 4\alpha^3 + 2\alpha^2 + 5\alpha + 4$	$\beta^9 = 4\alpha^3 + 5\alpha^2 + 6\alpha + 1$
$\beta^{10} = 5\alpha^4 + 7\alpha^3 + 6\alpha^2 + 7\alpha + 2$	$\beta^{11} = 7\alpha^4 + \alpha^3 + 7\alpha^2 + 2\alpha + 2$	$\beta^{12} = 5\alpha^4 + 2\alpha^3 + 5\alpha + 5$
$\beta^{13} = 3\alpha^4 + 7\alpha^3 + 2\alpha^2 + 4$	$\beta^{14} = \alpha^4 + 7\alpha^3 + 6\alpha + 3$	$\beta^{15} = 3\alpha^3 + 3$
$\beta^{16} = 4\alpha^4 + \alpha^2$	$\beta^{17} = 7\alpha^3 + 4\alpha^2 + 3\alpha + 4$	$\beta^{18} = \alpha^4 + 4\alpha^3 + 6\alpha^2 + 4\alpha + 1$
$\beta^{19} = 5\alpha^4 + 5\alpha^3 + \alpha^2 + 2\alpha + 5$	$\beta = 4\alpha^4 + 2\alpha^3 + 2\alpha^2 + 3\alpha + 3$	$\beta^{21} = \alpha^4 + 2\alpha^3 + 3\alpha^2 + 6\alpha + 5$
$\beta^{22} = 3\alpha^4 + 4\alpha^3 + \alpha^2 + \alpha + 3$	$\beta^{23} = 7\alpha^4 + 4\alpha^3 + 2\alpha^2 + 3\alpha + 6$	$\beta^{24} = 2\alpha^4 + 3\alpha^3 + 4\alpha^2 + 6\alpha$
$\beta^{25} = \alpha^4 + 2\alpha^3 + 5\alpha^2 + 4\alpha + 4$	$\beta^{26} = 2\alpha^4 + 2\alpha^3 + 3\alpha^2 + 7\alpha + 5$	$\beta^{27} = 3\alpha^4 + 7\alpha^3 + 5\alpha^2 + \alpha + 7$
$\beta^{28} = 4\alpha^4 + 3\alpha^2 + 7\alpha + 6$	$\beta^{29} = 6\alpha^4 + 5\alpha^3 + \alpha^2 + \alpha + 1$	$\beta^{30} = 5\alpha^3 + 3\alpha + 1$
$\beta^{31} = 1$		

Table 2.4: Elements of G_{31} over $GR(8, 5)$

2.1.2 Maximal Cyclic Subgroup with Computational Approach

Manually it is very time-consuming and difficult process to calculate the elements of the maximal cyclic subgroup of group of Galois ring units. So, it is essential to develop an algorithm that provides the maximal cyclic subgroup of any finite order within few seconds. The algorithm is designed in C# computer language is as follows, and program of this shown in appendix.

where $m_i(x)$ are minimal polynomials corresponding to each ξ^i for $i = 1, 2, 3, \dots, d$. The parity check matrix of the BCH code having generator polynomial $g(x)$ is of the form,

$$H = \begin{bmatrix} 1 & \xi^{c+1} & \xi^{2(c+1)} & \dots & \xi^{(n-1)(c+1)} \\ 1 & \xi^{c+2} & \xi^{2(c+2)} & \dots & \xi^{(n-1)(c+2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \xi^{c+d} & \xi^{2(c+d)} & \dots & \xi^{(n-1)(c+d)} \end{bmatrix}. \quad (2.2)$$

The following steps are performed to construct the generator polynomial for n length BCH codes over the Galois ring,

Step 1: Construct Maximal Cyclic Subgroup of order n .

Step 2: Find minimal polynomials corresponding to each designed distance.

Step 3: Take the least common multiple of all minimal polynomials.

Step 4: The output of the Step 3 is stated as generated polynomial $g(x)$ of BCH code.

Example 24 : Let $n = 15$, by using G_{15} as explained in Table 2.3 and equation 2.1 construct generator polynomials of BCH code for length 15 as,

designed distance	generator polynomial $g(x)$
5	$x^8 + 5x^7 + 3x^6 + 6x^5 + 7x^4 + 6x^3 + 2x^2 + 4x + 1$
6	$x^{10} + 6x^9 + x^8 + 6x^7 + 3x^5 + 7x^4 + 4x^3 + 7x^2 + 5x + 1$
8	$x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$

Table 2.5: generator polynomials for BCH code of length 15

Example 25 : Let the length of primitive BCH code over the Galois ring $GR(8, 5)$ is 31. Using the elements of G_{31} as calculated in Table 2.4, compute the generator polynomials corresponding to different designed distances as follows, for $d = 5$,

$$g(x) = x^{10} + 5x^9 + 7x^8 + 7x^6 + x^5 + 4x^4 + 5x^3 + 1. \quad (2.3)$$

For $d = 6$,

$$g(x) = x^{15} + 4x^{14} + 2x^{13} + 6x^{12} + 3x^{11} + 3x^{10} + 3x^9 + 5x^8 + 5x^7 + 5x^5 + 2x^4 + 3x^3 + 5x^2 + 3x + 7. \quad (2.4)$$

For $d = 10$,

$$\begin{aligned} g(x) = & x^{27} + 4x^{25} + 2x^{24} + 6x^{23} + x^{22} + 7x^{21} + 7x^{19} + 7x^{18} + 7x^{17} + 6x^{16} + \\ & 3x^{15} + 6x^{14} + 4x^{13} + 4x^{12} + 2x^{11} + 2x^{10} + 4x^9 + x^8 + 4x^7 + 6x^6 + \\ & 4x^5 + x^4 + 6x^2 + 5x + 7. \end{aligned} \quad (2.11)$$

For $d = 12$,

$$\begin{aligned} g(x) = & x^{33} + 7x^{32} + 3x^{30} + x^{29} + 5x^{28} + 5x^{27} + 5x^{26} + 6x^{25} + 6x^{24} + 3x^{23} + x^{22} \\ & + 6x^{21} + 3x^{20} + 4x^{19} + 6x^{18} + 6x^{17} + 2x^{16} + x^{15} + x^{14} + 3x^{13} + 4x^{12} + \\ & 5x^{11} + 3x^9 + 5x^8 + 5x^6 + 5x^5 + 4x^4 + 7x^2 + 7x + 7. \end{aligned} \quad (2.12)$$

For $d = 14$,

$$\begin{aligned} g(x) = & x^{39} + 5x^{38} + 7x^{37} + 3x^{36} + 2x^{35} + 5x^{34} + x^{33} + 4x^{32} + 7x^{31} + 2x^{30} + 6x^{29} \\ & + 3x^{28} + x^{27} + 2x^{26} + 5x^{25} + 2x^{24} + 7x^{23} + 7x^{22} + 4x^{21} + 6x^{20} + 6x^{19} \\ & + x^{17} + 6x^{16} + 4x^{15} + 6x^{14} + 4x^{12} + 5x^{11} + 4x^{10} + 5x^8 + 4x^7 + 4x^6 \\ & + x^5 + 6x^4 + 4x^2 + 7. \end{aligned} \quad (2.13)$$

For $d = 16$

$$\begin{aligned} g(x) = & x^{45} + 2x^{44} + 7x^{43} + x^{42} + x^{41} + x^{40} + 4x^{39} + 4x^{38} + 5x^{37} + 3x^{36} + 4x^{35} \\ & + 2x^{34} + 4x^{33} + 4x^{32} + 3x^{31} + 4x^{30} + 7x^{29} + 3x^{28} + 5x^{26} + 2x^{25} + x^{24} \\ & + 4x^{23} + 4x^{22} + 3x^{21} + 6x^{20} + 7x^{19} + 2x^{18} + 2x^{17} + x^{16} + 7x^{15} + 3x^{14} \\ & + 5x^{12} + 6x^{11} + 6x^{10} + 3x^9 + 7x^8 + 7x^7 + 3x^6 + 3x^4 + 5x^2 \\ & + 6x + 7. \end{aligned} \quad (2.14)$$

For $d = 32$,

$$\begin{aligned}
g(x) = & x^{62} + x^{61} + x^{60} + x^{59} + x^{58} + x^{57} + x^{56} + x^{55} + x^{54} + x^{53} + x^{52} + x^{51} \\
& + x^{50} + x^{49} + x^{48} + x^{47} + x^{46} + x^{45} + x^{44} + x^{43} + x^{42} + x^{41} + x^{40} + x^{39} \\
& + x^{38} + x^{37} + x^{36} + x^{35} + x^{34} + x^{33} + x^{32} + x^{31} + x^{30} + x^{29} + x^{28} + x^{27} \\
& + x^{26} + x^{25} + x^{24} + x^{23} + x^{22} + x^{21} + x^{20} + x^{19} + x^{18} + x^{17} + x^{16} + x^{15} \\
& + x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 \\
& + x^2 + x + 1
\end{aligned} \tag{2.18}$$

Example 27 : Generator polynomials of BCH code for length 127 over G_{127} corresponding to all possible designed distance as follows, for $d = 5$ the generator polynomial is,

$$g(x) = x^{14} + 2x^{13} + 7x^{12} + 2x^{11} + 7x^{10} + 6x^8 + 6x^7 + x^6 + 7x^5 + x^4 + 3x^3 + 7x^2 + 2x + 1. \tag{2.19}$$

For $d = 6$,

$$\begin{aligned}
g(x) = & x^{21} + 6x^{20} + 5x^{19} + 6x^{17} + 5x^{16} + 6x^{14} + x^{13} + x^{12} + 4x^{10} + 4x^9 + 2x^8 + 2x^7 \\
& + 2x^6 + 7x^4 + 5x^3 + x + 7.
\end{aligned} \tag{2.20}$$

For $d = 8$,

$$\begin{aligned}
g(x) = & x^{28} + 7x^{27} + 6x^{25} + 2x^{24} + x^{23} + 4x^{22} + x^{21} + 4x^{20} + 6x^{19} + x^{18} + 2x^{17} + 5x^{16} \\
& + 5x^{14} + 3x^{13} + x^{12} + 5x^{11} + 2x^{10} + 4x^9 + 3x^8 + 6x^6 + 5x^5 + 5x^4 + 3x^3 \\
& + 5x^2 + x + 1.
\end{aligned} \tag{2.21}$$

For $d = 20$,

$$\begin{aligned}
g(x) = & x^{63} + 5x^{62} + 7x^{61} + 5x^{60} + 5x^{58} + 4x^{57} + 2x^{56} + 3x^{55} + 2x^{52} + 2x^{51} + 7x^{50} + 2x^{49} \\
& + 6x^{48} + 6x^{47} + 5x^{46} + 5x^{44} + 4x^{43} + 7x^{42} + 6x^{41} + 3x^{40} + 2x^{39} + 4x^{38} + 2x^{37} \\
& + 7x^{36} + 5x^{35} + 2x^{33} + 4x^{32} + 3x^{31} + 6x^{30} + 5x^{29} + 5x^{28} + 5x^{27} + 4x^{26} + x^{24} \\
& + 4x^{23} + 3x^{22} + 2x^{21} + x^{20} + 5x^{19} + 2x^{18} + 2x^{17} + 4x^{15} + 4x^{14} + 5x^{13} + 4x^{12} \\
& + x^{11} + 6x^{10} + 5x^9 + 6x^8 + 4x^7 + 4x^6 + 2x^5 + 7x^4 + 3x^3 + 7x^2 + 7x + 7. \quad (2.26)
\end{aligned}$$

For $d = 22$,

$$\begin{aligned}
g(x) = & x^{70} + 4x^{68} + 6x^{67} + 2x^{66} + 2x^{65} + x^{64} + 5x^{63} + 3x^{62} + 6x^{61} + 3x^{60} + 3x^{59} \\
& + 4x^{58} + x^{57} + 4x^{56} + 5x^{55} + 2x^{53} + 7x^{52} + 6x^{51} + 2x^{50} + 7x^{48} + x^{47} \\
& + 6x^{46} + 6x^{45} + 7x^{44} + 5x^{43} + 5x^{42} + 6x^{41} + 7x^{40} + 3x^{39} + 2x^{38} + \\
& 2x^{37} + x^{36} + 5x^{35} + x^{34} + 5x^{33} + 6x^{32} + 7x^{31} + 7x^{30} + 3x^{29} + 5x^{28} + \\
& 3x^{27} + 4x^{26} + x^{25} + 7x^{24} + 2x^{23} + 3x^{21} + 3x^{20} + 4x^{19} + 7x^{18} + x^{17} + \\
& x^{15} + x^{13} + 6x^{12} + 2x^{11} + 7x^{10} + 3x^9 + 4x^8 + x^7 + 2x^6 + 4x^5 + x^4 + \\
& 7x^3 + 6x^2 + 4x + 1. \quad (2.27)
\end{aligned}$$

For $d = 24$,

$$\begin{aligned}
g(x) = & x^{77} + 6x^{76} + x^{75} + 4x^{73} + 3x^{72} + 2x^{71} + 4x^{69} + 3x^{68} + 6x^{66} + 7x^{65} \\
& + x^{64} + 4x^{63} + 4x^{62} + 3x^{60} + 5x^{59} + 3x^{58} + 3x^{57} + 3x^{56} + 4x^{55} \\
& + 2x^{54} + 4x^{53} + 6x^{52} + x^{51} + 2x^{50} + 6x^{49} + x^{48} + 7x^{47} + 4x^{46} \\
& + 3x^{45} + 6x^{44} + 3x^{43} + 5x^{42} + 4x^{41} + 6x^{40} + 6x^{39} + 3x^{38} + 3x^{37} \\
& + 7x^{36} + x^{35} + x^{33} + 4x^{32} + 6x^{29} + x^{28} + 7x^{27} + 5x^{26} + 2x^{25} \\
& + 2x^{24} + 2x^{23} + 6x^{22} + 4x^{21} + 2x^{20} + 2x^{19} + 5x^{18} + x^{17} + 4x^{16} \\
& + 7x^{15} + 3x^{14} + 7x^{13} + 4x^{12} + x^{11} + x^{10} + x^9 + 6x^8 + 5x^6 \\
& + 5x^5 + 4x^4 + x^3 + 5x^2 + 3x + 7. \quad (2.28)
\end{aligned}$$

For $d = 32$,

$$\begin{aligned}
g(x) = & x^{98} + 6x^{97} + 5x^{96} + 6x^{95} + 6x^{94} + 4x^{93} + 7x^{92} + 6x^{91} + 2x^{90} + \\
& 2x^{89} + 6x^{88} + 6x^{87} + x^{86} + x^{85} + 5x^{83} + 3x^{82} + 2x^{81} + 3x^{80} + x^{79} \\
& + 6x^{78} + 3x^{77} + 2x^{77} + 2x^{76} + 7x^{75} + 7x^{74} + 5x^{73} + 4x^{72} + 6x^{71} \\
& + 2x^{70} + 6x^{69} + x^{68} + x^{66} + 3x^{65} + 7x^{64} + 3x^{63} + 6x^{62} + 3x^{61} + \\
& 2x^{60} + 2x^{59} + x^{58} + 5x^{57} + 2x^{55} + 4x^{54} + 4x^{53} + 3x^{52} + x^{51} + x^{50} \\
& + 7x^{49} + 4x^{48} + 2x^{47} + 3x^{46} + 3x^{44} + 6x^{43} + x^{41} + 6x^{37} + 2x^{36} \\
& + x^{35} + x^{34} + 2x^{33} + 6x^{32} + 4x^{31} + x^{30} + 7x^{29} + 4x^{28} + 4x^{27} + \\
& 2x^{26} + 2x^{25} + 4x^{23} + 3x^{22} + 3x^{21} + 5x^{19} + 4x^{18} + 3x^{17} + 2x^{16} + \\
& 6x^{15} + x^{14} + 2x^{12} + x^{11} + x^{10} + 7x^9 + 6x^8 + 6x^7 + 2x^6 + 5x^5 + \\
& 2x^4 + 3x^3 + 4x^2 + 1.
\end{aligned} \tag{2.31}$$

For $d = 44$,

$$\begin{aligned}
g(x) = & x^{105} + x^{104} + x^{103} + 6x^{102} + 2x^{101} + x^{100} + 4x^{98} + x^{97} + 4x^{96} + 4x^{95} \\
& + 6x^{94} + 5x^{92} + 7x^{91} + 4x^{90} + x^{89} + x^{88} + x^{87} + 3x^{87} + 3x^{86} + x^{84} \\
& + 2x^{81} + 6x^{80} + 3x^{79} + 2x^{78} + 5x^{76} + 3x^{75} + 7x^{74} + 5x^{72} + 4x^{71} + x^{70} \\
& + 5x^{69} + 5x^{68} + 7x^{67} + 6x^{66} + 4x^{65} + 2x^{62} + 6x^{59} + 2x^{58} + 5x^{57} + 5x^{56} \\
& + x^{54} + 3x^{53} + 3x^{52} + 5x^{51} + 4x^{50} + 7x^{49} + 4x^{47} + x^{46} + 3x^{44} + 5x^{41} \\
& + 5x^{40} + 3x^{39} + 3x^{38} + 3x^{37} + x^{36} + 3x^{35} + 7x^{33} + 2x^{32} + 5x^{31} + 4x^{30} \\
& + 2x^{29} + 4x^{28} + 5x^{27} + 2x^{26} + 6x^{25} + x^{24} + 5x^{23} + 4x^{22} + 5x^{21} + 3x^{19} \\
& + 7x^{18} + 5x^{18} + 5x^{17} + 4x^{16} + 5x^{15} + x^{13} + x^{11} + 7x^{10} + x^9 + 2x^8 + 2x^7 \\
& + 5x^5 + 2x^4 + 3x^3 + 6x^2 + 5x + 7.
\end{aligned} \tag{2.32}$$

For $d = 64$,

$$\begin{aligned}
g(x) = & x^{126} + x^{125} + x^{124} + x^{123} + x^{122} + x^{121} + x^{120} + x^{119} + x^{118} + x^{117} \\
& + x^{116} + x^{115} + x^{114} + x^{113} + x^{112} + x^{111} + x^{110} + x^{109} + x^{108} + x^{107} \\
& + x^{106} + x^{105} + x^{104} + x^{103} + x^{102} + x^{101} + x^{100} + x^{99} + x^{98} + x^{97} + x^{96} \\
& + x^{95} + x^{94} + x^{93} + x^{92} + x^{91} + x^{90} + x^{89} + x^{88} + x^{87} + x^{86} + x^{85} + x^{84} \\
& + x^{83} + x^{82} + x^{81} + x^{80} + x^{79} + x^{78} + x^{77} + x^{76} + x^{75} + x^{74} + x^{73} + x^{72} \\
& + x^{71} + x^{70} + x^{69} + x^{68} + x^{67} + x^{66} + x^{65} + x^{64} + x^{63} + x^{62} + x^{61} + x^{60} \\
& + x^{59} + x^{58} + x^{57} + x^{56} + x^{55} + x^{54} + x^{53} + x^{52} + x^{51} + x^{50} + x^{49} + x^{48} \\
& + x^{47} + x^{46} + x^{45} + x^{43} + x^{42} + x^{41} + x^{40} + x^{39} + x^{38} + x^{37} + x^{36} + x^{35} \\
& + x^{34} + x^{33} + x^{32} + x^{31} + x^{30} + x^{29} + x^{28} + x^{27} + x^{26} + x^{25} + x^{24} + x^{23} \\
& + x^{22} + x^{21} + x^{20} + x^{19} + x^{18} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^{11} \\
& + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1.
\end{aligned} \tag{2.35}$$

Example 28 : Compute the generator polynomial of BCH code for length 255 over the Galois ring $GR(8, 8)$. By using the basic irreducible polynomial $f(x) = x^8 + 5x^4 + x^3 + 3x^2 + 3$ and G_{255} generator polynomials for designed distance $d = 111$,

For designed distance $d = 119$,

$$\begin{aligned}
g(x) = & x^{242} + 4x^{241} + 6x^{240} + x^{239} + 5x^{238} + 6x^{237} + 3x^{236} + 6x^{234} + 4x^{233} + 6x^{232} + 7x^{231} \\
& + 7x^{230} + 5x^{229} + 5x^{228} + 4x^{227} + 5x^{226} + 7x^{225} + 3x^{223} + 6x^{222} + 6x^{221} + 6x^{220} \\
& + 4x^{219} + 4x^{217} + 6x^{216} + x^{215} + 6x^{214} + x^{213} + 2x^{211} + 4x^{210} + 3x^{209} + 5x^{207} \\
& + 4x^{206} + 7x^{205} + 2x^{204} + x^{203} + 4x^{202} + x^{201} + x^{200} + 5x^{199} + x^{197} + 4x^{196} \\
& + 6x^{195} + x^{194} + 6x^{193} + 7x^{192} + x^{191} + 4x^{190} + 4x^{189} + 5x^{188} + 4x^{187} + 4x^{186} \\
& + 3x^{185} + 5x^{181} + 5x^{179} + 5x^{178} + x^{176} + 3x^{174} + 5x^{173} + 5x^{172} + x^{171} + 6x^{170} \\
& + 3x^{169} + 7x^{168} + 5x^{167} + 7x^{166} + x^{165} + 2x^{164} + 6x^{163} + x^{162} + 3x^{161} + 5x^{157} \\
& + x^{155} + 3x^{153} + 6x^{149} + 6x^{148} + 6x^{147} + 7x^{146} + x^{145} + 4x^{144} + 5x^{143} + x^{142} \\
& + 4x^{141} + 6x^{140} + 6x^{138} + 2x^{137} + 4x^{136} + x^{134} + 4x^{133} + 6x^{132} + 6x^{131} + x^{130} \\
& + x^{129} + 2x^{128} + 3x^{127} + 3x^{126} + 6x^{125} + 6x^{123} + 3x^{122} + 3x^{121} + 4x^{120} + x^{119} \\
& + 3x^{118} + x^{117} + 3x^{116} + x^{115} + x^{114} + 7x^{113} + 6x^{112} + 6x^{111} + x^{110} + 4x^{109} \\
& + 2x^{108} + 3x^{107} + 6x^{106} + 5x^{105} + 5x^{104} + 2x^{103} + 6x^{102} + 7x^{101} + 3x^{100} + 5x^{99} \\
& + 5x^{98} + 3x^{97} + 6x^{96} + x^{95} + 3x^{93} + x^{92} + 5x^{91} + 2x^{90} + 2x^{88} + x^{87} + 5x^{86} \\
& + 4x^{84} + 2x^{82} + 6x^{80} + 5x^{79} + x^{78} + 6x^{77} + 4x^{76} + 7x^{75} + 7x^{74} + 7x^{73} + x^{72} \\
& + 3x^{71} + 6x^{69} + 7x^{68} + x^{67} + 4x^{66} + 7x^{65} + 3x^{64} + 5x^{63} + 3x^{62} + x^{61} + 7x^{60} \\
& + 6x^{59} + 5x^{58} + 2x^{57} + 6x^{56} + 2x^{55} + 7x^{54} + 6x^{53} + 7x^{52} + 5x^{51} + x^{50} + 2x^{49} \\
& + 3x^{48} + 5x^{47} + 3x^{46} + 5x^{45} + 4x^{43} + 4x^{42} + 2x^{41} + 5x^{40} + 4x^{39} + x^{38} + 7x^{37} \\
& + 4x^{35} + 4x^{34} + 3x^{33} + 2x^{32} + 3x^{31} + 5x^{30} + x^{29} + 5x^{26} + 2x^{24} + 7x^{23} + 5x^{22} \\
& + 6x^{20} + 6x^{19} + 5x^{17} + 4x^{16} + x^{15} + 2x^{14} + 2x^{13} + 4x^{12} + 4x^{11} + 6x^{10} + 2x^9 \\
& + 7x^8 + 6x^7 + x^6 + 5x^5 + 4x^4 + x^2 + x + 1.
\end{aligned} \tag{2.37}$$

Example 29 : The generator polynomial of BCH code for length 511 over Galois ring $GR(8, 9)$ with designed distance 73. By using the basic irreducible polynomial, we get generator poly-

Example 30 : The generator polynomial of BCH code for length 1023 over Galois ring $GR(8, 10)$. By using the basic irreducible polynomial $f(x) = x^{10} + x^3 + 1$ and designed distance $d = 33$, generator polynomials as,

$$\begin{aligned}
g(x) = & x^{160} + x^{159} + 5x^{158} + 2x^{157} + 4x^{156} + 3x^{155} + 2x^{154} + 2x^{152} + 4x^{151} + 4x^{150} + 6x^{149} \\
& + 5x^{148} + 2x^{147} + 5x^{146} + 7x^{145} + 3x^{144} + 5x^{142} + 2x^{141} + 5x^{138} + x^{137} + 5x^{135} \\
& + 5x^{134} + x^{133} + 4x^{131} + 7x^{130} + 5x^{129} + 3x^{128} + 7x^{127} + 4x^{126} + 6x^{123} + 5x^{122} \\
& + 2x^{121} + 6x^{120} + x^{119} + 2x^{118} + 6x^{117} + 3x^{116} + 2x^{114} + x^{113} + 6x^{112} + 4x^{111} \\
& + 5x^{109} + x^{108} + 5x^{106} + x^{105} + 3x^{104} + 7x^{102} + 6x^{101} + 2x^{100} + 3x^{99} + 7x^{98} \\
& + 5x^{97} + 5x^{96} + 4x^{95} + 3x^{94} + x^{93} + 2x^{92} + 2x^{91} + 2x^{90} + x^{89} + 7x^{88} + 4x^{86} \\
& + x^{85} + 3x^{84} + x^{83} + 6x^{82} + x^{81} + 6x^{80} + 3x^{79} + 3x^{78} + 3x^{77} + 6x^{74} + 4x^{73} \\
& + 6x^{72} + 4x^{71} + 4x^{70} + 7x^{69} + 2x^{68} + 6x^{67} + 4x^{65} + 3x^{64} + 5x^{63} + 4x^{62} + 7x^{61} \\
& + 5x^{60} + x^{59} + 6x^{58} + x^{57} + 2x^{56} + x^{55} + x^{54} + 4x^{53} + 7x^{52} + 5x^{51} + 7x^{50} + 5x^{48} \\
& + 2x^{47} + x^{42} + 7x^{39} + 3x^{38} + 2x^{37} + 2x^{36} + 6x^{35} + 4x^{34} + 6x^{33} + 2x^{32} + 3x^{31} \\
& + 2x^{29} + 2x^{28} + 4x^{27} + 2x^{26} + x^{25} + 2x^{24} + 7x^{23} + x^{22} + 7x^{21} + 7x^{20} + x^{19} + x^{18} \\
& + 4x^{17} + 6x^{16} + 3x^{15} + x^{14} + 4x^{13} + 6x^{12} + 6x^{11} + 7x^{10} + 7x^9 + 4x^8 + 6x^7 + 5x^6 \\
& + 6x^5 + 2x^4 + 5x^3 + x^2 + 6x + 1.
\end{aligned} \tag{2.39}$$

2.2 Computationally Encoding of BCH Codes over Galois Ring

In this section, novel computational approach is introduced to calculate generator polynomial. The computational new scheme designed in computer language $C\#$. The explanation of an algorithm with a flow chart is given in Fig. 2.1. Thus, this new scheme for BCH codes over $GR(p^m, r)$ gives generator polynomial very fast.

the coefficient of x^0 . The computational procedure `Add(poly1, poly2)` adds two polynomials in a Galois ring $GR(p^k, m)$ and gives the sum of these polynomials. All calculation is done in \mathbb{Z}_{p^k} where p^k is used in the base of the variable in an algorithm. We have done this computational technique in computer language C#.

Description of Algorithm 1

In this algorithm, 'poly' is the input of basic irreducible polynomial, and q is a power of some prime p , d is designed distance, and n is the length of the code. Line 1 and line 2 compute the exponents of the root of the basic irreducible polynomial. Line 3 to line 6 calculates all exponents of the root of the basic irreducible polynomial until the output is 1. Line 7 computes the generator of the maximal cyclic subgroup. Line 8 to 11 compute the list of elements of the maximal cyclic subgroup. Line 12 computes all roots of minimal polynomials of BCH codes over Galois ring. Line 13 calculates all minimal polynomials over the Galois ring, and line 14 calculates all minimal polynomial over corresponding Galois field. Line 15 calculates the generator polynomial of BCH codes over the Galois ring by taking LCM of all minimal polynomials. Line 16 calculates the generator polynomial of BCH codes over the Galois field.

Algorithm 1

ComputeGeneratorPoly(poly, q,p,d,n)

Begin

1. `i := poly.MaxDegree`
2. `listAlphaPoly := GetPreviousAlphaPoly(i)`
3. `repeat:`
4. `poly := Mul(poly, " ")`
5. `listAlphaPoly.Add (SubstituteAndSimplyfy(poly, i))`
6. `until PolynomialIsConstant(poly)= false`
7. `generatorOfMCSG:= listAlphaPoly.Count / n;`
8. `for betaPower := 1 To n`
9. `betaPoly := listAlphaPoly [generatorOfMCSG*betaPower - 1]`
10. `MCSG.Add(betaPoly)`
11. `end loop`

```

9. end if
10. for coeffIndex := 0 To poly.MaxDegree
11. poly[coeffIndex]:= (poly[coeffIndex] % base + base)% base;
12. end loop
13. listAlphaPoly.Add(poly)
End

```

Add(poly1,poly2)

```

Begin
1. for i = 0 To poly2.MaxDegree
2. if (i >= poly1.MaxDegree) then
3. if (poly1.Length > i) then
4. poly1[i]:= poly1[i] + poly2[i];
5. else
6. for j = poly1.MaxDegree+1 To i
7. poly1[j]:= 0;
8. end loop
9. poly1[i]:= poly2[i]
10. end if
11. CONTINUE:
12. else
13. poly1[i]:= poly1[i] + poly2[i]
14. end if
15. end loop
16. return poly1

```

End

Sub(poly1,poly2)

```

Begin
1. for i = 0 To poly2.MaxDegree
2. if (i >= poly1.MaxDegree) then
3. if (poly1.Length > i) then

```



```
5. end loop
6. if p[0] = 1 then
7. return true
8. end if
9. return false
```

End

CalcRootsForAllMinPoly(p,d,n)

Begin

```
1. for i := 1 To d
2. tempBetaPowers := new List()
3. k := 0
4. power := 0
5. repeat:
6. power := (Pow(p,k++) * i)%n
7. if tempBetaPowers.Contains(power) then
8. break
9. else
10. tempBetaPowers.Add(power);
11. end if
12. until true
13. listRootsForAllMinPoly.Add(tempBetaPowers)
14. end loop
15. return listRootsForAllMinPoly
```

End

CalcAllMinPoly(listRootsForAllMinPoly,betaPoly,p)

Begin

```
1. i := 1
2. foreach MkRoots in listRootsForAllMinPoly
3. tempMkPoly:= CalcMk(MkRoots,bPoly))
4. listMinPolys.Add(tempMkPoly)
```

```

8. rowInd := cordInd.y + row
9. colInd := (cordInd.x + col) % n;
10.newCoef := mvPolys[sn][cordInd.y][cordInd.x] * mvPolyResult[row][col]
11.tempResult[rowInd][colInd] += newCoef
12. end if
13. end loop
14. end loop
15. end loop
16. CopyAndCleanArray(tempResult, mvPolyResult)
17. end loop
18. constPoly := -1
19. Mk := ZeroPolynomial
20. for x := 0 To mvPolyResult.Length
21. for beta := 1 To mvPolyResult[0].Length
22. if mvPolyResult[x][beta] != 0 then
23. constPoly[0] := mvPolyResult[x][beta]
24. testPoly := Mul(bPolys[beta - 1], constPoly)
25. tempPoly := Add(tempPoly, testPoly)
26. end if
27. end loop
28. if x > 0 then
29. constPoly[0] := mvPolyResult[x][0]
30. tempPoly := Add(tempPoly, constPoly)
31. end if
32. for coeff := 0 To tempPoly.MaxDegree
33. tempPoly[coeff] := (tempPoly[coeff] % q + q) % q
34. end loop
35. if tempPoly.MaxDegree = -1 AND x = mvPolyResult.Length-1 then
36. Mk := Add(Mk, )
37. else

```

Step 1: GetPreviousAlphaPoly(degreeOfPoly)

Here, the loop calculates all those powers α of which are less than the degree of the basic irreducible polynomial. For example, the degree of the basic irreducible polynomial is 8. Therefore output is $1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7$.

Step 2: SubstituteAndSimplify(poly, i, listAlphaPoly)

To solve input polynomial of degree m , take the degree of a polynomial on the left side of an equation, and the remaining terms shifted to the right side of an equation. Apply modulo q on the right side of an equation, where q is the power of some prime number. After that, it calculates the powers of α , while calculating the powers of α ; this function substitutes the previous value of α . If the exponent of α is greater than or equal to the degree of a basic irreducible polynomial, then simplify the polynomial with the help of modulo q operation. Here q is 8, and the degree of the polynomial is also 8. The output will be $\alpha^8 = -\alpha^4 - \alpha^3 - \alpha^2 - 1 \pmod{8} = 7\alpha^4 + 7\alpha^3 + 7\alpha + 7$, $\alpha^9 = 7\alpha^5 + 7\alpha^4 + 7\alpha^3 + 7\alpha^2 + 7\alpha$ and continue till we get 1.

Step 3: Add(poly1,poly2)

Here the addition of two polynomials 'poly1' and 'poly2' is performed. For loop, add the coefficients of same exponents and save the output in the new array then compute the final result. For example, while calculating the power of α and simplifying this procedure, add two polynomials.

Step 4: Sub(poly1,poly2)

This function returns the subtraction of two polynomials 'poly1' and 'poly2'. For loop, perform the subtraction of polynomials having the same powers. Subtract the coefficients and take modulo q , then save the result in the new array, is added to the negative coefficients of until we get the positive answer. For example, $\alpha^8 = -\alpha^4 - \alpha^3 - \alpha^2 - 1$, then take modulo $q = 8$. The output will be $\alpha^8 = 7\alpha^4 + 7\alpha^3 + 7\alpha + 7$.

Step 5: Mul(poly1,poly2)

We multiply two polynomials, as the coefficients of $\alpha^{\varsigma+\tau}$, where $\varsigma = 0, 1, 2, \dots, n$ and $\tau = 0, 1, 2, \dots, m$. The addition of the product of coefficients of the corresponding elements with power addition $\varsigma + \tau$. The product of the coefficients is then added into the output polynomial. For example, multiplication is performed during the calculation of minimal polynomials.

Step 6: PolynomialIsConstant(poly)

calculation of minimal polynomials, and it also used in the calculation of the maximal cyclic subgroup.

Step 13: LCMOfMinPoly(listOfGeneratorPoly)

Calculate the lcm of all minimal polynomials, which is the multiplication of all distinct minimal polynomials. Therefore the output for generator polynomial of BCH codes over Galois ring and corresponding Galois field are displayed.

2.3 Encoding of BCH Codes over Galois Field

The encoding of message is the process to add redundant bits in a message such that k length message is embedded in n length message. If during transmission n length message contains errors, then we can determine those errors and correct them. The encoding process helps us in the transmission of data securely.

How to encode a message

Any message u of k bits can be encoded by following steps:

- i) Write a message u into the form of polynomial $u(x)$.
- ii) Find the generating polynomial $g(x)$.
- iii) Encoded message $c(x) = u(x)g(x)$, where $c(x)$ is n length code polynomial.

Example 31 : To encode the message $w = 11010$ through the encoder $[15, 5]$ and designed distance $d = 7$, here $n = 15$ and $k = 5$.

Step 1: Message u into the form of a polynomial is $u(x) = 1 + x + x^3$.

Step 2: To find generator polynomial $g(x)$, firstly, construct the elements of Galois field $GF(16)$, which is generated by primitive polynomial $f(x) = x^4 + x + 1$. Assume that ξ is the primitive root of $h(x)$ then $\xi^4 + \xi + 1 = 0$ and $\xi^4 = \xi + 1$,

Hence the $g(x)$ of the BCH code is,

$$\begin{aligned}
 g(x) &= \text{lcm}\{m_i(x) : i = 1, 2, 3, \dots, 6\}, \\
 &= m_1(x) m_2(x) m_3(x), \\
 &= (x^4 + x + 1) (1 + x + x^2 + x^3 + x^4) (1 + x + x^2), \\
 &= 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}
 \end{aligned} \tag{2.43}$$

Step 3: Encoded message is $c(x) = u(x) g(x)$,

$$\begin{aligned}
 c(x) &= (1 + x + x^3) (1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}) \\
 &= 1 + x^5 + x^6 + x^7 + x^9 + x^{10} + x^{13}.
 \end{aligned}$$

$c = 100001110110010$ is desired encoded message of 15 length.

Remark 32 Code rate and error-correcting capability remains the same for codes in Galois rings $GR(p^n, m)$ and in corresponding Galois fields $GF(p^m)$. But the advantage of Galois ring is the number of codewords is greater than the codewords in the Galois field.

2.3.1 Construction of Generator Polynomial of BCH Codes over Galois Field

If we have a generator polynomial of BCH codes over the Galois ring, then there is no need to construct a generator polynomial of BCH codes over the Galois field, separately. Take modulo p operation to the coefficients of a generator polynomial of BCH codes over the Galois ring.

Example 33 : Suppose that the generator polynomial of BCH code of length 15 with designed distance $d = 5$ over Galois ring $GR(8, 4)$ is,

$$g(x) = x^8 + 5x^7 + 3x^6 + 6x^5 + 7x^4 + 6x^3 + 2x^2 + 4x + 1.$$

To calculate the generator polynomial of BCH code over corresponding Galois field $GF(2^4)$ with designed distance $d = 5$, take modulo 2 operation to the coefficients of $g(x)$. Therefore, $\overline{g(x)} = x^8 + x^7 + x^6 + x^4 + 1$.

Table 2.7 shows that the dimension of some primitive BCH code can be chosen corresponding to different designed distances. Similarly, we can find the dimension of primitive BCH using a computational approach corresponding to any length.

Chapter 3

BCH-Codes over Galois Ring and Galois Field: Computational Decoding Approach

In Coding theory, generally, codes are designed for the reliable transmission of data through noisy channels by using classical and more efficient algebraic techniques. There are many coding theory applications in different fields for example, in magnetic and optical recording, wireless and network communication systems. Error detection and error correction are primary goals in coding theory because, during data transmission, there is possible in most cases error must occur due to distortion, noise, interference. The main aim of coding theory is to design error control codes. To achieve the aim of coding theory, many mathematicians and engineers use different algebraic techniques to develop good codes as much as possible. In this chapter, the decoding of BCH codes over the Galois ring and the Galois field using the Barlekamp Massey algorithm with the computational approach are explained. It is very challenging to correct errors if encoded messages have large lengths. To deal with this problem, we have designed computational technique which corrects multiple errors for each length of the BCH code.

$$\begin{aligned}
\tilde{S}_1 &= (\gamma_1) + (\gamma_2) + \dots + (\gamma_\nu), \\
\tilde{S}_2 &= (\gamma_1)^2 + (\gamma_2)^2 + \dots + (\gamma_\nu)^2, \\
\tilde{S}_3 &= (\gamma_1)^3 + (\gamma_2)^3 + \dots + (\gamma_\nu)^3, \\
&\cdot \\
&\cdot \\
&\cdot \\
\tilde{S}_{2t} &= (\gamma_1)^{2t} + (\gamma_2)^{2t} + \dots + (\gamma_\nu)^{2t}.
\end{aligned} \tag{3.6}$$

We define error locator polynomial as,

$$\begin{aligned}
\sigma(x) &= (1 + \gamma_1 x)(1 + \gamma_2 x)(1 + \gamma_3 x) \dots (1 + \gamma_\nu x), \\
\sigma(x) &= \sigma_0 + \sigma_1 x + \sigma_2 x^2 + \dots + \sigma_\nu x^\nu,
\end{aligned} \tag{3.7}$$

here the coefficients of x are elementary symmetric functions, and defined as,

$$\begin{aligned}
\sigma_0 &= 1 \\
\sigma_1 &= \gamma_1 + \gamma_2 + \dots + \gamma_\nu \\
\sigma_2 &= \gamma_1 \gamma_2 + \gamma_2 \gamma_3 + \dots + \gamma_{\nu-1} \gamma_\nu \\
&\cdot \\
&\cdot \\
&\cdot \\
\sigma_\nu &= \gamma_1 \gamma_2 \gamma_3 \gamma_4 \dots \gamma_\nu
\end{aligned} \tag{3.8}$$

Remark 34 : *The inverses of roots of $\sigma(x)$ are error location numbers.*

We connect elementary symmetric functions with the syndromes by using the newton identities as follows,

polynomial.

Step 1: If $d_\mu = 0$, then

$$\sigma^{\mu+1}(x) = \sigma^\mu(x) \text{ and } l_{\mu+1} = l_\mu. \quad (3.9)$$

Step 2: If $d_\mu \neq 0$, then find $m < \mu$ such that $d_m \neq 0$, and $m - l_m$ has largest value in last column of the table, such that

$$\sigma^{\mu+1}(x) = \sigma^\mu(x) + d_\mu d_m^{-1} x^{(\mu-m)} \sigma^m(x), \quad (3.10)$$

$$l_{\mu+1} = \max(l_\mu, l_m + \mu - m). \quad (3.11)$$

Step 3: To find discrepancy use this formula,

$$d_{\mu+1} = S_{\mu+2} + \sigma_1^{(\mu+1)} S_{\mu+1} + \dots + \sigma_{l_{\mu+1}}^{(\mu+1)} S_{\mu+2-l_{\mu+1}}. \quad (3.12)$$

The polynomial $\sigma^{2t}(x)$ in the last row of the table is required error locator polynomial. Next, calculate roots of this polynomial and then take inverses of these roots, these inverses of roots are known as error location numbers. The powers of the error location numbers show error positions in received code. To correct these errors, subtract the error vector from the received vector. If we have to find the message word, then divide the corrected message polynomial by the generator polynomial $g(x)$.

Example 35 : Let us consider $[15, 5, 7]$ BCH code generated by $g(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$ and codeword $\tilde{c} = 100001110110010$ is sent through channel. The error may occur during transmission and received vector is $\tilde{r} = 110101110110011$.

Now find the error in the received vector. Choose $n = 15$, $k = 5$, $d = 7$ so $t = \lfloor \frac{d-1}{2} \rfloor = 3$.

$$GF(16) = \frac{\mathbb{Z}_2[X]}{\langle x^4 + x + 1 \rangle}, \quad (3.13)$$

Where $x^4 + x + 1$ is primitive polynomial, and let ξ be the primitive root of this polynomial, therefore, $1 + \xi + \xi^4 = 0$. The received vector in the form of a polynomial is ,

$$\tilde{r}(x) = 1 + x + x^3 + x^5 + x^6 + x^7 + x^9 + x^{10} + x^{13} + x^{14}$$

$\max\{0, 1\} = 1$.

For d_1 , put $\mu = 0$ in formula (3.16) we get, $d_1 = S_2 + \sigma_1^{(1)}S_1 = \xi^8 + \xi^4 \cdot \xi^4 = 0$, here $\sigma_1^{(1)}$ is coefficient of x in $\sigma^1(x)$.

Since $d_1 = 0$, therefore $\sigma^2(x) = \sigma^1(x)$ and $l_2 = l_1$, which implies that $\sigma^2(x) = 1 + \xi^4x$ and $l_2 = 1$.

To calculate d_2 , put $\mu = 1$ in formula (3.16) then, $d_2 = S_3 + \sigma_1^{(2)}S_2 + \sigma_2^{(2)}S_1 = (1 + \xi^2 + \xi^3) + \xi^4 \cdot \xi^8 + 0 \cdot \xi^4 = 1 + \xi^2 + \xi^3 + \xi^{12}$, here $\sigma_1^{(2)}$ is coefficient of x in $\sigma^{(2)}(x)$ and $\sigma_2^{(2)}$ is coefficient of x^2 in $\sigma^{(2)}(x)$, from the table of Galois field $\xi^{12} = 1 + \xi + \xi^2 + \xi^3$, therefore, $d_2 = 1 + \xi^2 + \xi^3 + 1 + \xi + \xi^2 + \xi^3 = \xi \neq 0$.

Since $d_2 \neq 0$, we find $\sigma^3(x)$ by using the formula given by (3.15), put $\mu = 2$ and $m = 0$ because $d_0 \neq 0$ and $0 - l_0 = 0$ is largest in last column of the table.

Therefore, $\sigma^3(x) = \sigma^2(x) + d_2d_0^{-1}x^{2-0}\sigma^0(x) = (1 + \xi^4x) + (\xi)(\xi^4)^{-1}x^2(1) = 1 + \xi^4x + (\xi)(\xi^{11}) = 1 + \xi^4x + \xi^{12}x$.

Also put $\mu = 2$ in formula given by (3.17) we get $l_3 = \max\{l_2, l_0 + 2 - 0\} = \max\{1, 2\} = 2$ and $\mu - l_3 = 3 - l_3 = 3 - 2 = 1$.

To compute d_3 , put $\mu = 2$ in formula 3.16 then $d_3 = S_4 + \sigma_1^{(3)}S_3 + \sigma_2^{(3)}S_2 = \xi + \xi^4 \cdot \xi^{13} + \xi^{12} \cdot \xi^8$, which implies that, $d_3 = 0$ (by using the table of Galois field). Since $d_3 = 0$ so $\sigma^4(x) = \sigma^3(x)$ and $l_4 = l_3 = 2$ so $\mu - l_\mu = 4 - l_4 = 4 - 2 = 2$.

$\sigma^4(x) = 1 + \xi^4x + \xi^{12}x$

For d_4 , substitute $\mu = 3$ in formula given by (3.15). Therefore, $d_4 = S_5 + \sigma_1^{(4)}S_4 + \sigma_2^{(4)}S_3 = 0 + \xi^4 \cdot \xi + \xi^{12} \cdot \xi^{13} = \xi^5 + \xi^{10} = 1$. Since $d_4 \neq 0$, therefore we calculate $\sigma^5(x)$,

put $\mu = 4$, and $m = 2$ in formula given by (3.15),

$\sigma^5(x) = \sigma^4(x) + d_4d_2^{-1}x^{(4-2)}\sigma^2(x) = 1 + \xi^4x + \xi^{12}x + (1)(\xi)^{-1}x^2(1 + \xi^4x) = 1 + \xi^4x + \xi^{12}x + \xi^{14}x^2(1 + \xi^4x)$, which implies that, $\sigma^5(x) = 1 + \xi^4x + \xi^5x^2 + \xi^3x^3$ and $l_5 = \max\{l_4, l_2 + 4 - 2\} = \max\{2, 1 + 4 - 2\} = 3$.

To find d_5 , put $\mu = 4$ in formula given by (3.15), then $d_5 = S_6 + \sigma_1^{(5)}S_5 + \sigma_2^{(5)}S_4 + \sigma_3^{(5)}S_3 = \xi^{11} + \xi^4 \cdot 0 + \xi^5 \cdot \xi + \xi^3 \cdot \xi^{13} = 0$. Since $d_5 = 0$, therefore $\sigma^6(x) = \sigma^5(x)$ which implies that $\sigma^6(x) = 1 + \xi^4x + \xi^5x^2 + \xi^3x^3$.

3.1.2 Decoding of BCH Codes over Binary Field

To decode binary BCH codes by using the Barlekamp Massey algorithm, we start the algorithm by initial conditions explained in Table 3.3. There are total t steps required to find error locator polynomial.

μ	$\sigma^\mu(x)$	d_μ	l_μ	$2\mu - l_\mu$
$\frac{-1}{2}$	1	1	0	-1
0	1	S_1	0	0

Table 3.3: Initial conditions for \mathbb{Z}_2

Barlekamp Massey Algorithm for Binary BCH Codes

For decoding of the binary BCH codes, following steps are required,

Step 1: Start with $\mu = -1/2$.

Step 2: If $d_\mu = 0$ then $\sigma^{(\mu+1)}(x) = \sigma^{(\mu)}(x)$.

Step 3: If $d_\mu \neq 0$, then find $m < \mu$ such that $2m - l_m$ is large as possible in last column of the table and $d_m \neq 0$, such that

$$\sigma^{(\mu+1)}(x) = \sigma^{(\mu)}(x) + d_\mu d_m^{-1} x^{2(\mu-m)} \sigma^{(m)}(x), \quad (3.19)$$

$$d_{\mu+1} = S_{2\mu+3} + \sigma_1^{(\mu+1)} S_{2\mu+2} + \sigma_2^{(\mu+1)} S_{2\mu+1} + \dots + \sigma_{l_{\mu+1}}^{(\mu+1)} S_{2\mu+3-l_{\mu+1}}, \quad (3.20)$$

$$l_{\mu+1} = \deg\left(\sigma^{(\mu+1)}(x)\right). \quad (3.21)$$

The polynomial $\sigma^t(x)$ in last row of the table is required error locator polynomial.

Example 36 : *If apply this algorithm to example 35 then error locator polynomial is computed after 3 steps as follows,*

Step 1: *Calculate the syndrome.*

Step 2: *Apply Barlekamp Massey algorithm with initial conditions, substitute $\mu = 0$, $m = \frac{-1}{2}$ in the formula given by (3.19),*

If we compare Table 3.2 and Table 3.4, we get $\sigma^3(x)$, and $\sigma^6(x)$ are same error locator polynomial. By using this algorithm, half steps are required to find error locator polynomial.

Remark 37 : *If the degree of error locator polynomial is greater than t , then there are more than t error occurs.*

Remark 38 : *The computation required for binary BCH code is one half of the calculation required for non-binary BCH code.*

Remark 39 : *If the number of errors in the received vector is less than t , then it is unnecessary to find t steps for error locator polynomial.*

3.2 Decoding of BCH over Galois Field: Computational Approach

It is very time-consuming to decode large length BCH code manually over a higher-order field. To sort out this time-consuming effort, we developed the program in computer language which help to decode the message very efficiently. Following algorithms gives output for decoding of BCH codes very fast and program of this algorithm is shown in appendix.

3.2.2 Algorithm 2 for Decoding of BCH Codes

Algorithm2

Algorithm 2: Decoding of BCH Codes over Galois Field

Input :

Barlekamp-Massey Table `barlekampMasseyTable`
List of Alpha Polynomials `alphaPolyList`
Length of Code `n`
Value `p`
Receive Vector `recVec`

Output :

Error Vector `errVec`
Corrected Code `corCode`

```
1. tempPoly ← barlekampMasseyTable.MaxSigma
2. errVec ← 0

3. for i ← 1 to alphaPolyList.Count
4.   foreach term in tempPoly
5.     if term.alphaDegree = i then
6.       Substitute term with alphaPolyList[i]
7.       term ← term.Degree (mod n)
8.       term ← term.Coefficient (mod p)
9.     end if
10.  end loop
11. errorPoly.Coefficient[0] ← tempPoly.Coefficient[0]
12. for index ← 1 to tempPoly.Coefficient.Count
13.   if tempPoly.Coefficient[index] != 0 then
14.     errorPoly ← errorPoly.Add(alphaPolyList[index])
15.   end if
16. end loop
17. errorPoly ← errVec.Coefficients (mod p)
18. if errorPoly is Zero then
19.   errVec ← errVec.Add("x^" + (-i + n).ToString())
20. end if
21. end loop
22. corCode ← recVec.Subtract(errVec)
23. corCode ← corCode.Coefficients (mod p)
```

4.png

Algorithm to calculate all syndromes

Calculation

Algorithm: Calculation of Syndromes

Input :

Receive Vector `recVec`

List of Alpha Polynomials `alphaPolyList`

Output :

List of Syndrom `syndromList`

```

1. for syndromCount ← 1 to 2*t
2.   for i ← 1 to recVec.length
3.     polyIndex ← i*syndromCount (mod n)
4.     if degree != 0 then
5.       syndromPoly ← alphaPolyList[polyIndex]
6.       syndromPoly ← syndromPoly.Coefficients (mod p)
7.       syndromList.Add(syndromPoly)
8.     end if
9.   end loop
10. end loop

```

6.png

3.3 Decoding of RS and BCH-Codes over Galois Ring

RS codes are non-binary cyclic codes. Recently, RS codes have too many applications in disk drives, satellite communication, compact disk player, DVDs, and two-dimensional bar codes.

3.3.1 Reed-Solomon Codes over \mathbb{Z}_{p^m}

The parameters of RS codes over \mathbb{Z}_{p^m} are $(p-1, p-d, d)$. If $m=1$ and $d=2t+1$ where $t \leq \lfloor \frac{n-1}{2} \rfloor$, then Reed Solomon code correct t errors. Let $\zeta \in \mathbb{Z}_{p^m}$ be a primitive element. Suppose that $n=p-1$ and $m \in \mathbb{Z}^+$ such that $(n, m) = 1$. Then according to Blake [2] the parity check matrix H is defined as:

$$H = \begin{bmatrix} 1 & \zeta^m & \zeta^{2m} & \dots & \zeta^{(n-1)m} \\ 1 & \zeta^{m+1} & \zeta^{2(m+1)} & \dots & \zeta^{(n-1)(m+1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta^{m+d-2} & \zeta^{2(m+d-2)} & \dots & \zeta^{(n-1)(m+d-2)} \end{bmatrix} \quad (3.25)$$

Theorem 40 [2]: *The null space of the parity check matrix H over \mathbb{Z}_{p^m} is $p-1$ length code C with dimension $p-d$ and minimum distance d .*

Where τ represents number of errors which are occurred by the channel.

To find $\overline{\sigma}_1, \overline{\sigma}_2, \dots, \overline{\sigma}_\tau$, we solve the following equations over ring \mathcal{R} .

$$S_{i+\tau} + S_{i+\tau-1}\overline{\sigma}_1 + S_{i+\tau-2}\overline{\sigma}_2 + \dots + \widetilde{S}_j\overline{\sigma}_\tau = 0, \quad i = 1, 2, \dots, 2t - \tau, \quad (3.28)$$

where $S_1, S_2, S_3, S_4, \dots, S_{2t}$ are syndromes. Equations given by (3.28) can be solved by using the modified Barlekamp Massey algorithm. If error magnitudes are unit elements in the ring \mathcal{R} , then the solution is unique. It is an iterative algorithm because at μ th step.

Find l_μ values $\overline{\sigma}_i^{(\mu)}$ such that, $\mu - l_\mu$ equations holds with l_μ possibly small and $\overline{\sigma}_0^{(\mu)} = 1$.

$$\begin{aligned} S_\mu \overline{\sigma}_0^{(\mu)} + S_{\mu-1} \overline{\sigma}_1^{(\mu)} + \dots + S_{\mu-l_\mu} \overline{\sigma}_{l_\mu}^{(\mu)} &= 0 \\ S_{\mu-1} \overline{\sigma}_0^{(\mu)} + S_{\mu-2} \overline{\sigma}_1^{(\mu)} + \dots + S_{\mu-l_\mu-1} \overline{\sigma}_{l_\mu}^{(\mu)} &= 0 \\ &\vdots \\ &\vdots \\ &\vdots \\ S_{l_{\mu+1}} \overline{\sigma}_0^{(\mu)} + S_{l_\mu} \overline{\sigma}_1^{(\mu)} + \dots + S_1 \overline{\sigma}_{l_\mu}^{(\mu)} &= 0. \end{aligned} \quad (3.29)$$

The output at final step is:

$$\overline{\sigma}^{(\mu)}(x) = \overline{\sigma}_0^{(\mu)} + \overline{\sigma}_1^{(\mu)}x + \overline{\sigma}_{l_\mu}^{(\mu)}x^{l_\mu}, \quad (3.30)$$

The n th discrepancy,

$$d_\mu = S_{\mu+1} \overline{\sigma}_0^{(\mu)} + S_\mu \overline{\sigma}_1^{(\mu)} + \dots + S_{\mu+1-l_\mu} \overline{\sigma}_{l_\mu}^{(\mu)}. \quad (3.31)$$

Step 3: There is one extra step to find error location number over ring rather than over field because in the ring \mathcal{R} the solution of the equation given by (3.28) is not unique. The reciprocal of polynomial $\overline{\sigma}^{2t}(x)$ (output of Barlekamp Massey algorithm) is $\rho(x)$, which may or may not be the correct error locator polynomial. To compute error location numbers, we initially find the roots x_1, x_2, \dots, x_v of $\rho(x)$, then select $z_0 = \alpha^0, z_1 = \alpha^1, z_2 = \alpha^2, \dots, z_{n-1} = \alpha^{n-1}$ such that,

equations of (3.29) and discrepancy satisfy following condition,

$$d_\mu + a \cdot d_m = 0 \quad (3.38)$$

and

$$l_m = l_{\mu+1} - (\mu - m). \quad (3.39)$$

Theorem 44 [15]: Let $\overline{\sigma^{(\mu)}(x)}$ be a solution at μ th step and $\overline{\sigma^{(m)}(x)}$ be the existing minimal solutions, for $\mu > m \geq 1$, such that $m - l_m$ is the greatest value in last column of the table and $d_\mu - y \cdot d_m = 0$ have solutions in y . Further, suppose that $\overline{\sigma^{(\mu)}(x)}$ is modified by the following method. If $d_\mu = 0$, then

$$\overline{\sigma^{(\mu+1)}(x)} = \overline{\sigma^{(\mu)}(x)} \text{ and } l_{\mu+1} = l_\mu. \quad (3.40)$$

If $d_\mu \neq 0$, then

$$\overline{\sigma^{(\mu+1)}(x)} = \overline{\sigma^{(\mu)}(x)} - y \cdot (x)^{\mu-m} \cdot \overline{\sigma^{(m)}(x)} \quad (3.41)$$

and

$$l_{\mu+1} = \max\{l_\mu, l_m + \mu - m\}. \quad (3.42)$$

If there does not exist any solution $\overline{D^{(\mu+1)}(x)}$ with degree less than $\max\{l_\mu, l_m - m + \mu\}$, and the coefficient of smallest exponent of x in $\overline{D^{(\mu+1)}(x)} - \overline{\sigma^{(\mu)}(x)}$ is zero divisor in the ring \mathcal{R} , then at $(\mu + 1)$ th stage $\overline{\sigma^{(\mu+1)}(x)}$ is the minimal polynomial solution.

3.3.3 Modified Berlekamp Massey Algorithm

Modified Berlekamp Massey Algorithm is designed to decode BCH or RS codes over $GR(p^m, r)$ or \mathbb{Z}_p^m , respectively. We take syndromes as an input and get elementary symmetric functions as an output that satisfies equations given by (3.29) for minimum v . We start this algorithm by following initial conditions as in [15], are as follows,

$\overline{\sigma^{(-1)}(x)} = 1$, $l_{-1} = 0$, $d_{-1} = 1$, $\overline{\sigma^{(0)}(x)} = 1$, $l_0 = 0$, and $d_0 = \tilde{S}_1$. Here, \tilde{S}_1 is first non-zero syndrome. Further steps of the algorithm is as follows,

Step 1: $0 \rightarrow \mu$.

Step 2: If $d_\mu = 0$, then $\overline{\sigma^{(\mu)}(x)} \rightarrow \overline{\sigma^{(\mu+1)}(x)}$, $l_\mu \rightarrow l_{\mu+1}$ and then go to Step 5.

Step 3: If $d_\mu \neq 0$, then find a $m < \mu$, such that $d_\mu - y d_m = 0$ has solution in y , $m - l_m$ has

Step 1: To calculate syndrome we use,

$$S = r \cdot H^T = (17 \ 20 \ 4 \ 107). \quad (3.48)$$

Step 2: Calculation of error locator polynomial which satisfy equation given by (3.28) by using, Modified Berlekamp Massey algorithm, $\overline{\sigma^{2t}(x)}$ is calculated as follows,

μ	$\overline{\sigma^\mu(x)}$	d_μ	l_μ	$\mu - l_\mu$
-1	1	1	0	-1
0	1	17	0	0
1	$1 + 104x$	94	1	0
2	$1 + 70x$	73	1	1
3	$1 + x + 84x^2$	97	2	1
4	$1 + 61x + 49x^2$	-	-	-

Table 3.5: Error locator polynomial

From Table 3.5 required error locator polynomial is $\overline{\sigma^4(x)} = 49x^2 + 61x + 1$.

Step 3: The reciprocal of $\overline{\sigma^4(x)}$ is $\overline{\rho(x)} = x^2 + 61x + 49$. By substituting $x \in \{0, 1, 2, 3, 4, \dots, 120\}$, we have $x_1 = 32$ and $x_2 = 28$ are roots of $\overline{\rho(x)}$. Next select those $z_i \in \{\beta^0, \beta^1, \beta^2, \beta^3, \beta^4, \beta^5, \beta^6, \beta^7, \beta^8, \beta^9\}$ such that $z_i - x_i$ are zero-divisors in \mathbb{Z}_{121} . Therefore, $z_1 = \beta^5$ and $z_2 = \beta^9$ are correct error location numbers, and powers of β represent error positions in the received vector. Hence error occurs in the received vector at 5th and 9th position.

Step 4: Firstly, calculate correct elementary symmetric function as,

$$(x - z_1)(x - z_2) = (x - 32)(x - 28) = x^2 + 61x + 49, \text{ which implies that } \overline{\sigma_1} = 61 \text{ and } \overline{\sigma_2} = 49.$$

Apply Forney's procedure to calculate error magnitudes,

$$y_1 = \frac{\overline{\sigma_{1,0}} \cdot \tilde{S}_2 + \overline{\sigma_{1,1}} \tilde{S}_1}{\overline{\sigma_{1,0}} \cdot z_1^2 + \overline{\sigma_{1,1}} \cdot z_1}. \quad (3.49)$$

If $r \cdot H^t = 0$ then there is no error occurs in received vector.

Step 1: Calculation of Syndrome as,

$$S = r \cdot H^t = (3 \ 3x \ 3 \ 3). \quad (3.54)$$

Step 2: Calculate the error locator polynomial which satisfy the equation given by (3.28), using the Modified Berlekamp Massey algorithm,

μ	$\overline{\sigma^\mu(Z)}$	d_μ	l_μ	$\mu - l_\mu$
-1	1	1	0	-1
0	1	3	0	0
1	$1 + 6Z$	$3x$	1	0
2	$1 + (6 + 8x)Z$	$3x$	1	1
3	$1 + (5 + 8x)Z + 3Z^2$	$6x$	2	1
4	$1 + (3 + 8x)Z + (2x)Z^2$	-	-	-

Table 3.6: Error locator polynomial of BCH codes

Therefore, the error locator polynomial from the last row of Table 3.6,

$$\overline{\sigma^4(Z)} = 1 + (3 + 8x)Z + (2x)Z^2.$$

Step 3: For Calculation of correct error location number, the reciprocal of $\overline{\sigma^4(Z)}$ is $\overline{\rho(Z)}$, which is correct error locator polynomial, $\overline{\rho(Z)} = Z^2 + (3 + 8x)Z + 2x$. The roots of $\overline{\rho(Z)}$ in Galois ring $GR(9, 2)$ are $Z_1 = 5 + 8x$ and $Z_2 = 1 + 2x$. Select those elements X_i of G_8 which satisfy, $X_i - Z_i$ are zero-divisors in the ring $GR(9, 2)$. Therefore, $X_1 = \beta^1 = 2 + 8x$ and $X_2 = \beta^6 = 7 + 5x$. Hence, β^1 and β^6 are correct error location numbers and powers of β shows error positions. Therefore, the error occurs at 1st and 6th position. To find elementary symmetric function, compute $(Z - X_1)(Z - X_2) = (Z - (2 + 8x))(Z - (7 + 5x)) = Z^2 + (5x)Z + (6 + 8x)$ which implies that $\overline{\sigma_1} = 5x$ and $\overline{\sigma_2} = 6 + 8x$.

Step 4: To calculate error magnitudes, apply Forney's procedure to syndrome and elementary symmetric function. Therefore, we get error magnitude $Y_1=3$ and $Y_2 = 6$. Hence the error vector $e = (0 \ 3 \ 0 \ 0 \ 0 \ 0 \ 6 \ 0)$ and the corrected codeword $v = r - e = (0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0)$.

Chapter 4

Symmetric Block Cipher and BCH Codes: An Image Encryption Application

In the modern world, the security of the digital image is vital due to the occurrence of communication digital products over open networks frequently. Accelerated advancement of digital data exchange, the importance of information security in the transmission of data through the communication channel, and its storage has emerged. Multiple usages of the images in the security agencies and the industries, the security of the confidential image data from unauthorized access is emergent and vital. In this chapter, Error-Correcting Codes, particularly Bose Chaudhuri Hocquenghem codes over the Galois field, have been used in image encryption using a modified Advanced Encryption Standard algorithm. The BCH codes over the Galois field are used in the mixed column operation and round key addition steps of the AES algorithm. The detailed analyses of the proposed scheme and its comparison to the original AES algorithm are given in this chapter. We designed a novel technique for the construction of components of the block cipher.

cations. On the bases of the AES key expansion image encryption scheme is explained in [28]. Error-correcting codes, particularly BCH codes, are helpful in reducing the rate of a decryption failure. In [39], the decoding algorithm of the BCH code and design a constant-time version of the BCH decoding algorithm is analyzed. The modified AES algorithm for image and text encryption, which is based on bit permutation instead of mixed column operation, is presented in [8]. But we used BCH codes in mixed column operation and round key addition in the AES algorithm for image encryption.

4.2 Advanced Encryption Standard Algorithm

Advanced Encryption Standard is a symmetric block cipher system that uses exchange or replaces network. According to the required key length and block length of AES can be varied. For an iteration of 10, 12, and 14 rounds, three different key length schedules 128, 192, and 256 are used respectively. Key size determines the level of security. AES consist of the round function that's composed four different byte-oriented transformation. Key expand turns, and round change are the three main aspects of the AES algorithm. The collection of three layers add-round key layer, non-linear layer, and the linear mixture layer is the transformation of each round.

4.2.1 Substitute Byte Transformation

It is a non-linear byte substitution that applies independently on each byte of state using S-Box. This operation explains how each byte of the state matrix substitutes with another byte of the Substitution-Box. S-Box contains 256 elements. There are different techniques to construct S-Box.

4.2.2 Shift Rows Transformation

In this Transformation, the bytes of the rows of the current state matrix is left-shifted cyclically. Row 0 is unchanged, and the first row is shifted one byte to the left. In the second row, there are two bytes left shift are performed. Similarly, apply to the remaining rows.

BCH codes. The modulo multiplication is performed over the Galois field $GF(2^8)$.

Step 6: Take the analysis of encrypted image and compare it with the original image.

4.3.2 Construction of Round Keys using BCH Codes

The construction technique of round keys followed by the binary representation of the generator polynomials of BCH codes over $GF(2^7)$ for different designed distances. Convert each generator polynomial to its binary representation form of length 128-bits. If it is not 128-bits, add check bits on the left-hand side to make 128-bits long. Then convert the BCH of length 128 into 16-bytes. These 16 bytes string serves as a round key. Key 1 is derived from the generator polynomial of BCH code $[n = 127, k = 1]$ with the designed distance 65. By using the proposed technique, we get key 1 as follows,

Key 1: 127 255 255 255 255 255 255 255 255 255 255 255 255 255 255,

We shall construct next key by using BCH code of length 127 over Galois field $GF(2^7)$ with the designed distance 60. Then convert the coefficients of generator polynomial which are in descending order into the block of 8 bits,

00000000 10101011 00110001 00010110 10011100 10000100 10100100 11011011
10001111 01000001 10101000 11001011 10110000 11101011 11001111 10111111.

Convert each byte into decimal form so hence key 2 as,

Key 2: 0 171 49 22 156 132 164 219 143 65 168 203 176 235 207 191.

Similarly, construct all keys by using BCH codes over Galois field by changing the designed distance.

Key 3: 01 101 123 192 163 7 249 56 40 16 229 154 109 22 187

Key 4: 0 0 3 146 27 208 157 120 3 122 83 250 137 174 174 43

Key 5: 0 0 0 5 16 109 174 23 166 30 82 12 96 106 78 41

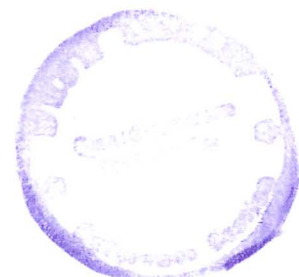
Key 6: 0 0 0 0 13 83 6 214 191 219 200 87 71 25 231 13

Key 7: 0 0 0 0 0 25 161 99 10 46 46 13 22 111 12 93

Key 8: 0 0 0 0 0 0 41 19 31 9 172 122 28 6 238 111

Key 9: 0 0 0 0 0 0 0 65 218 145 157 158 251 54 166 153

Key 10: 0 0 0 0 0 0 0 0 244 132 85 24 185 88 42 31



form. The matrix is as follows,

$$A = \begin{pmatrix} 233 & 68 & 73 & 57 \\ 21 & 97 & 9 & 161 \\ 244 & 112 & 4 & 116 \\ 164 & 207 & 49 & 35 \end{pmatrix} \quad (4.3)$$

This is the required matrix which is used in the AES algorithm in mixed column transformation step for the security of data.

4.3.4 Application of BCH Codes in Security of Image Data

Digital images are vulnerable to unauthorized access while in transmission over a communication channel. Streaming digital images also require high network bandwidth for transmission. For effective image transmission over the internet, security issues must be considered. Recently, We apply the proposed scheme in image encryption of size 256×256 .

The encrypted image of Lena using the proposed scheme is given in Fig. 4.2 and statistical analyses of an encrypted image with the original AES algorithm and a modified AES algorithm is shown in Table 4.1.



Fig. 4.1: Lena original Image

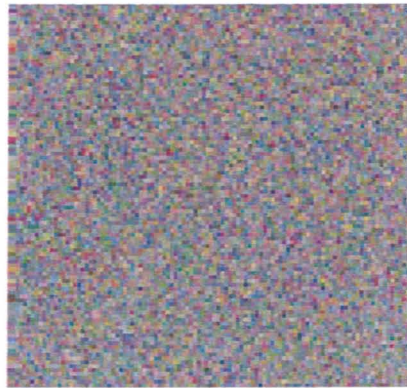


Fig. 4.2: Lena encrypted image

4.4.3 Energy

This analysis computes the energy of the encrypted images by applying S-Boxes. It gives the sum of squared elements in the grey level co-occurrence matrix,

$$\sum_{l,m} p(l-m)^2, \quad (4.6)$$

where $p(l, m)$ is the number of gray-level co-occurrence matrices.

4.4.4 Homogeneity

In homogeneity, the gray level co-occurrence matrix explains the proficiency of arrangements of pixel brightness results in tabular form. The closeness of the distribution in the Gray level co-occurrence matrix to its diagonal is measured through the homogeneity analysis. If the homogeneity is small as much possible, then encryption is better. The following formula measures the homogeneity,

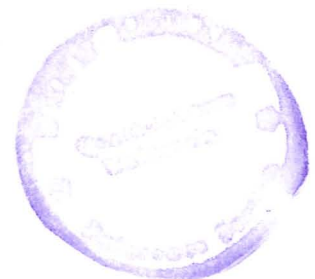
$$H = \sum_{l,m} \frac{n(l, m)}{1 + |l - m|}. \quad (4.7)$$

4.4.5 Entropy

Information entropy measures the disorder, which is created by the encryption process. Entropy measures the strength of the encryption technique. An encryption technique is good if it has more disorder and randomness. Entropy is defined as,

$$e = - \sum_{i=1}^n p(x_i) \log_b p(x_i), \quad (4.8)$$

where $P(x_i)$ contains the histogram counts. Entropy must be close to the 8 for better image quality.



of BCH code corrects the error. It means that if there is an error in encrypted data during transmission, then there is no secure decryption. So, we apply the decoder of BCH codes, which correct the errors in encrypted data and then decrypt the data and attain the original message.

4.4.6 Histogram Analyses

Histogram analysis is used to see how much encryption procedure changes test image compared to the encrypted image. For good encryption, the histogram of the ciphered image should have a uniform distribution indicating that the anticipated scheme can resist statistical attacks. The histograms of the ciphered images are appreciably uniform and are quite dissimilar from the test images. The suggested encryption technique has fulfilled all the test image features and has convoluted the statistical bond between the test image and its cipher image.

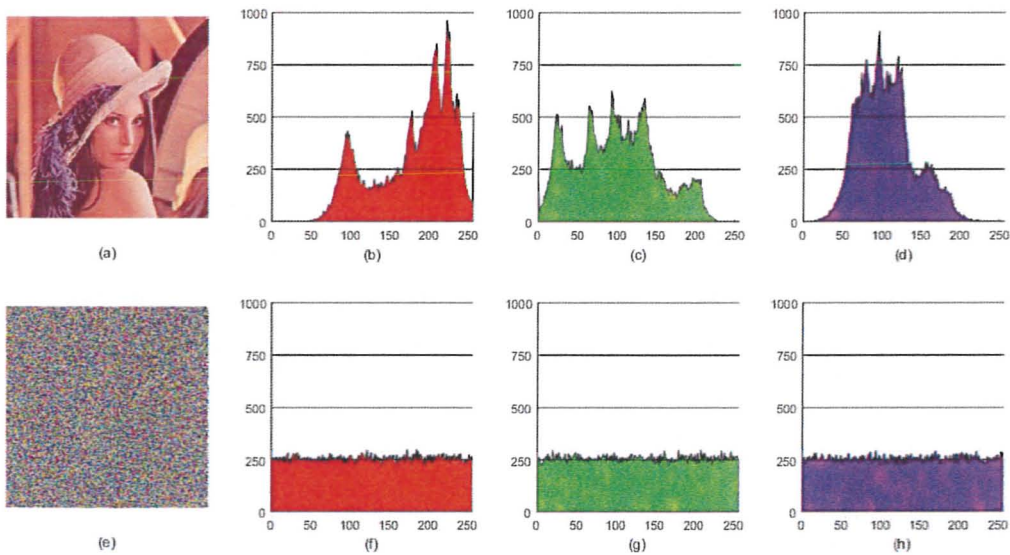


Fig. 4.3: Histogram analysis of original and encrypted mage

Fig. 4.3 shows that the histogram analyses of the original and encrypted images by the proposed encryption algorithm through different channels (Red, Green, Blue). The histogram

Chapter 5

Symmetric Block Cipher and BCH Codes: A Text Encryption Application

Accelerated adventures in computer science and technology has made digital technology, a need of the day. Academicians, researchers, and technologists are anxious to share their secret data through the communication channel along with its security. With the fast advancement of digital data exchange, security information becomes much important in transmission and data storage. Due to the extreme use of texts in security agencies and industry, it is essential to keep the confidential text data from unauthorized access. In chapter 4, we have utilized BCH codes in image encryption and used one mixed column matrix in each round of modified AES algorithm, but in this chapter, BCH codes are used for the security of text data by utilizing ten different mixed column matrices for each round in modified AES algorithm. The proposed scheme improves the mixed column operation by using MDS matrices and round key addition steps of the modified AES algorithm using BCH codes over the Galois field for better encryption performance and security of data. Using the avalanche effect for text encryption and the most popular test NIST, it is ensuring that the proposed scheme for ciphertext is more beneficial, and results of security analyses are given in this chapter.

$$\begin{aligned}
g(x) = & x^{242} + x^{239} + x^{238} + x^{236} + x^{231} + x^{230} + x^{229} + x^{228} + x^{226} + x^{225} + x^{223} + x^{215} \\
& + x^{213} + x^{209} + x^{207} + x^{205} + x^{203} + x^{201} + x^{200} + x^{199} + x^{197} + x^{194} + x^{192} + x^{191} \\
& + x^{188} + x^{185} + x^{181} + x^{179} + x^{178} + x^{176} + x^{174} + x^{173} + x^{172} + x^{171} + x^{169} + x^{168} \\
& + x^{167} + x^{166} + x^{165} + x^{162} + x^{161} + x^{157} + x^{155} + x^{153} + x^{146} + x^{145} + x^{143} + x^{142} \\
& + x^{134} + x^{130} + x^{129} + x^{127} + x^{126} + x^{122} + x^{121} + x^{119} + x^{118} + x^{117} + x^{116} + x^{115} \\
& + x^{114} + 7x^{113} + x^{110} + x^{107} + x^{105} + x^{104} + x^{101} + x^{100} + x^{99} + x^{98} + x^{97} + x^{96} \\
& + x^{95} + x^{93} + x^{92} + x^{91} + x^{87} + x^{86} + x^{79} + x^{78} + x^{75} + x^{74} + x^{73} + x^{72} + x^{71} \\
& + x^{68} + x^{67} + x^{65} + x^{64} + x^{63} + x^{62} + x^{61} + x^{60} + x^{58} + x^{54} + x^{52} + x^{51} + x^{50} \\
& + x^{48} + x^{47} + x^{46} + x^{45} + x^{40} + x^{38} + x^{37} + x^{33} + x^{31} + x^{30} + x^{29} + x^{26} + x^{23} \\
& + x^{22} + x^{17} + x^{15} + x^8 + x^6 + x^5 + x^2 + x + 1.
\end{aligned} \tag{5.3}$$

Divide $(x^{n-k+i-1})$ for $i = 1$ by $g(x)$ and obtain a remainder polynomial then split to the coefficients of remainder polynomial into the block of 8 bits and write these blocks into 4×4 matrix after converting it into decimal form. Dividing the polynomial (x^{255-13}) by $g(x)$ we get matrix as,

$$A = \begin{pmatrix} 233 & 68 & 73 & 57 \\ 21 & 97 & 9 & 161 \\ 244 & 112 & 4 & 116 \\ 164 & 207 & 49 & 35 \end{pmatrix}. \tag{5.4}$$

If we divide the polynomial (x^{243}) by $g(x)$ then we get,

$$A_1 = \begin{pmatrix} 221 & 104 & 182 & 89 \\ 110 & 180 & 91 & 44 \\ 234 & 50 & 155 & 207 \\ 168 & 113 & 251 & 190 \end{pmatrix}. \tag{5.5}$$

If we divide the polynomial (x^{248}) by $g(x)$ then we obtain,

$$A_6 = \begin{pmatrix} 246 & 119 & 96 & 56 \\ 123 & 59 & 176 & 28 \\ 203 & 234 & 184 & 54 \\ 101 & 245 & 92 & 27 \end{pmatrix}. \quad (5.10)$$

If we divide the polynomial (x^{249}) by $g(x)$ then we get,

$$A_7 = \begin{pmatrix} 153 & 101 & 108 & 223 \\ 76 & 178 & 182 & 111 \\ 38 & 89 & 91 & 55 \\ 19 & 45 & 173 & 155 \end{pmatrix}. \quad (5.11)$$

If we divide the polynomial (x^{250}) by $g(x)$ then we get,

$$A_8 = \begin{pmatrix} 248 & 84 & 26 & 157 \\ 132 & 126 & 23 & 83 \\ 66 & 63 & 11 & 233 \\ 33 & 31 & 133 & 244 \end{pmatrix}. \quad (5.12)$$

If we divide the polynomial (x^{251}) by $g(x)$ then we get matrix as,

$$A_9 = \begin{pmatrix} 136 & 52 & 201 & 200 \\ 68 & 26 & 100 & 228 \\ 34 & 13 & 50 & 114 \\ 153 & 50 & 80 & 241 \end{pmatrix}. \quad (5.13)$$

5.2 Construction of Secret Keys using BCH Codes

The construction technique of round keys followed by the binary representation of the generator polynomials of BCH codes over the Galois field $GF(2^m)$ for different designed distances. Convert each generator polynomial to its binary representation form for 128 length key.

For Key 3, Construct the generator polynomial for designed distance $d = 44$,

$$\begin{aligned}
g(x) = & x^{105} + x^{104} + x^{103} + x^{100} + x^{97} + x^{92} + x^{91} + x^{89} + x^{88} + x^{87} + x^{87} + x^{86} \\
& + x^{84} + x^{79} + x^{76} + x^{75} + x^{74} + x^{72} + x^{70} + x^{69} + x^{68} + x^{67} + x^{57} + x^{56} \\
& + x^{54} + x^{53} + x^{52} + x^{51} + x^{49} + x^{46} + x^{44} + x^{41} + x^{40} + x^{39} + x^{38} + x^{37} \\
& + x^{36} + x^{35} + x^{33} + x^{31} + x^{27} + x^{24} + x^{23} + x^{21} + x^{19} + x^{18} + x^{18} + x^{17} \\
& + x^{15} + x^{13} + x^{11} + x^{10} + x^9 + x^5 + x^3 + x + 1.
\end{aligned} \tag{5.16}$$

Therefore, **Key 3**: 0 0 0 5 16 109 174 23 166 30 82 12 96 106 78 41.

For Key 4, calculate the generator polynomial for designed distance $d = 32$,

$$\begin{aligned}
g(x) = & x^{98} + x^{96} + x^{92} + x^{86} + x^{85} + x^{83} + x^{82} + x^{80} + x^{79} + x^{77} + x^{75} + x^{74} + x^{73} \\
& + x^{68} + x^{66} + x^{65} + x^{64} + x^{63} + x^{61} + x^{58} + x^{57} + x^{52} + x^{51} + x^{50} + x^{49} \\
& + x^{46} + x^{44} + x^{41} + x^{35} + x^{34} + x^{30} + x^{29} + x^{22} + x^{21} + x^{19} + x^{17} \\
& + x^{14} + x^{11} + x^{10} + x^9 + x^5 + x^3 + 1.
\end{aligned} \tag{5.17}$$

Therefore, **Key 4**: 0 0 0 0 13 83 6 214 191 219 200 87 71 25 231 13.

For Key 5, find the generator polynomial for designed distance $d = 30$,

$$\begin{aligned}
g(x) = & x^{91} + x^{90} + x^{88} + x^{86} + x^{84} + x^{81} + x^{80} + x^{74} + x^{73} + x^{71} + x^{70} + x^{68} + x^{66} \\
& + x^{65} + x^{63} + x^{61} + x^{60} + x^{59} + x^{58} + x^{57} + x^{56} + x^{55} + x^{54} + x^{52} + x^{51} \\
& + x^{49} + x^{48} + x^{47} + x^{46} + x^{43} + x^{38} + 5x^{36} + x^{34} + x^{33} + x^{32} + x^{30} + x^{26} \\
& + x^{25} + x^{24} + x^{20} + x^{19} + x^{16} + x^{15} + x^{14} + x^{13} + x^{10} + x^9 + x^8 \\
& + x^3 + x^2 + 1,
\end{aligned} \tag{5.18}$$

which implies that, **Key 5**: 0 0 0 0 0 25 161 99 10 46 46 13 22 111 12 93.

For Key 10, calculate the generator polynomial for designed distance $d = 16$,

$$\begin{aligned}
 g(x) = & x^{56} + x^{53} + x^{51} + x^{49} + x^{48} + x^{46} + x^{45} + x^{44} + x^{43} + x^{42} + x^{41} + x^{40} \\
 & + x^{39} + x^{35} + x^{32} + x^{28} + x^{25} + x^{24} + x^{23} + x^{20} + x^{17} + x^{16} + x^{13} + x^{11} \\
 & + x^{10} + x^4 + 1.
 \end{aligned} \tag{5.23}$$

Hence, **Key 10**: 0 0 0 0 0 0 0 0 2 146 195 22 238 162 115.

5.3 Application of BCH Codes in Text Data

BCH codes are utilized in text encryption application. First-time BCH codes can be used as a secret key in any encryption scheme. BCH codes are also used in mixed column matrices.

Encryption Scheme

AES algorithm is modified for text encryption using the BCH codes as follows,

Step 1: Convert 128 bits plain text data into 16 data bytes and write these 16 bytes in a 4×4 state matrix.

Step 2: Construct keys by using the BCH codes of length 128 by taking different designed distances over the Galois field, which are used as round keys. Key 0 is used in round 0, and key 1 is used in round 1, apply all 10 different keys in 10 rounds.

Step 3: Now, the entries of the current state matrix are substituted with the AES S-Box entries.

Step 4: Now perform the circular shift on each row of the current state matrix. Row 0 is shifted 0 bytes left, row 1 is shifted 1 byte left, row 2 is shifted 2 bytes left, and row 3 shifted 3 bytes left.

Step 5: Now, the current state matrix is multiplied in each round with the different mix column MDS matrix constructed by using BCH codes. The multiplication is modulo multiplication over the Galois field $GF(2^8)$.

Step 6: Apply the analysis on ciphered text and compare it with the original AES algorithm.

Example 54 : Suppose that we want to encrypt plain text **ONE TWO NINE ONE** by

Round 7
<i>Plain Text</i> : b9 68 05 7b 52 7b 66 53 7e d5 16 36 1e 05 88 6a
<i>Cipher Text</i> : ea 09 88 ac 87 a5 42 da 43 e7 5d d9 3d 5f 90 2d
Round 8
<i>Plain Text</i> : ea 09 88 ac 87 a5 42 da 43 e7 5d d9 3d 5f 90 2d
<i>Cipher Text</i> : e5 64 5e 14 5b 3b 55 a8 2d d3 55 20 3e 63 2d a8
Round 9
<i>Plain Text</i> : e5 64 5e 14 5b 3b 55 a8 2d d3 55 20 3e 63 2d a8
<i>Cipher Text</i> : 15 35 97 5c ba e2 71 5f 04 44 71 d8 80 6d 53 62
Round 10
<i>Plain Text</i> : 15 35 97 5c ba e2 71 5f 04 44 71 d8 80 6d 53 62
<i>CipherText</i> : 59 98 a6 aa f4 1b ed 4a f2 3e 1a 0c db 78 01 12

5.4 Text Encryption Analyses

We encrypt text data by using the proposed scheme, which is based on BCH codes. Proposed scheme is analyzed by standard analyses. These analyses are avalanche effect, ciphertext attack, known-plaintext attack and NIST test.

5.4.1 Avalanche Effect

Every encryption method has its strong and weak arguments. To apply the appropriate method in a specific application, we must require identifying the weakness and strengths. Therefore, the analyses of these methods are critically compulsory. Every encryption algorithm has ensured the property, a little bit change in either the plaintext or the key, should produce a huge change in the ciphertext. The Avalanche Effect is defined, if we single bit change in the key or in the plain text then it gives us several bits change in the ciphertext. The strength of the proposed algorithm is estimated using Avalanche Effect due to single-bit variation in plaintext keeping

We test ciphertext by changing in a single bit in original Key and compute all 10 rounds then compare the output with cipher text constructed with the original key.

Round	Cipher Texts with Original Key and with Single Bit Change	Bits	%
0	4f 4e 65 20 54 77 6f 20 4e 69 6e 65 20 4f 4e 65 4f 4e 65 20 54 77 6f 20 4e 69 6e 65 20 4f 4e 65	1	
1	7e 00 f4 04 7f cf 3e e8 f7 d5 fd 8a 8c c7 9d df 9b 27 8f 51 7f cf 3e e8 f7 d5 fd 8a 8c c7 9d df	69	53.91
2	26 b5 54 c4 71 3e 5b 4b 77 d8 ee bf a8 45 01 f7 a5 77 39 4c 90 0f 54 5f 80 b8 b0 44 77 51 84 d9	63	49.22
3	56 49 9d 94 e6 8e 39 fe 6d 17 f1 b5 3b 10 f2 71 0f 0d bf 74 5e 5c 96 3f 9f 01 f6 92 fd cc bf fe	61	47.66
4	04 36 c3 00 3d 75 34 d0 51 f3 e2 23 cf 86 4a 62 2d 4a 37 54 2a cc be b9 ce 83 19 f8 1a 98 b3 14	74	57.81
5	6d d6 19 77 23 a6 e1 d8 6a 7e 43 01 7c 30 57 11 04 a9 64 f2 e7 2e c0 39 45 07 0d 00 a1 67 25 44	65	50.78
6	b9 68 05 7b 52 7b 66 53 7e d5 16 36 1e 05 88 6a e1 f7 de c7 7d 51 15 5d e3 65 09 3f ce 5e 30 4e	65	50.78
7	ea 09 88 ac 87 a5 42 da 43 e7 5d d9 3d 5f 90 2d e1 f7 de c7 7d 51 3c 0f 26 fd 38 db 29 6e 78 b8	66	51.56
8	e5 64 5e 14 5b 3b 55 a8 2d d3 55 20 3e 63 2d a8 b6 9e cb 5a 4c 14 e2 fa 18 bc 35 6f ec 7b 1e 7c	67	52.34
9	15 35 97 5c ba e2 71 5f 04 44 71 d8 80 6d 53 62 85 33 82 5f 61 2b 68 9b ab 15 39 68 7a ee e7 d5	58	45.31
10	59 98 a6 aa f4 1b ed 4a f2 3e 1a 0c db 78 01 12 97 f1 12 03 ef 59 94 cf 62 2a 81 d7 cc 2d e7 36	61	47.66

Table 5.2: Avalanche effect by changing one bit in secret key

In Table 5.1, we apply the avalanche effect by altering a single bit in plain text keeping unchanged in secret key, and then performing all the rounds of the proposed AES algorithm.

comparison to limited numbers for the random sequence. The excursion of this random walk should be close to zero. The excursion of this random walk for non-random sequences will be far from zero. The purpose of the cumulative sums test is to decide whether the sum of the partial sequences happening in the tested sequence is too small or too large.

Statistical Test	<i>Decision</i>
The Discrete Fourier Transform (spectral) test	Passed
The Non-overlapping template matching test	Passed
The approximate entropy test	Passed
The Cumulative Sums (forward) test	Passed
The Cumulative Sums (reverse) test	Passed

Table 5.3: NIST statistical test passed for text encryption

5.4.3 Ciphertext Attack

If the cryptanalyst knows the ciphertext and the encryption techniques, but does not have the private key to decrypt the ciphertext. If the brute force attack is applied to ciphertext, it will not be helpful to attain plain text. If the key size is too large it will take many years to decrypt the ciphered message. Therefore, even if the analyst attains the original message, then the data's will be insufficient at that time.

5.4.4 Known Plaintext Attack

Given that the cryptanalyst knows the encryption algorithm, ciphertext, and one or more ciphertext-plaintext pairs designed by the private key. Since the implementation generates a different ciphertext for a similar message due to the different designed distance of BCH codes, known-plaintext attack cannot harm.

Chapter 6

A Nonlinear Component Design in Symmetric Block Cipher with Image Encryption Application

In chapter 4, we encrypt the image using the BCH codes and the modified AES algorithm, but the security layer was missing in that algorithm. In this chapter, we constructed a nonlinear component (S-Boxes) in symmetric block cipher using the Galois ring and the Galois field and apply those S-Boxes in data security. An S-Box is based on Boolean functions, which are essential in the foundation of symmetric cryptographic systems. The Boolean functions are used for S-Box designing in the block cipher. Boolean functions with optimal nonlinearity and upright cryptographic stuff play a significant role in block ciphers' design. To analyze the security of the image encryption, some standard analyses are performed.

6.1 Construction of Nonlinear Component using Galois Ring and Galois Field

The S-Box is the crucial component used in several cryptosystems. It works in the way of substituting several blocks of bits for a completely unlike set of output bits. So the substitution shows a confused association among input and output bits of the substitution box. When used

<i>Exp</i>	<i>Polynomial</i>	<i>Exp</i>	<i>Polynomial</i>	<i>Exp</i>	<i>Polynomial</i>	<i>Exp</i>	<i>Polynomial</i>
1	00010000	30	40573013	59	51443017	88	16545373
2	00037505	31	74436313	60	26541144	89	4502214
3	75065707	32	30444347	61	71717214	90	26431410
4	27625706	33	50777444	62	46055235	91	74133547
5	32516562	34	67225333	63	10127701	92	2635357
6	70227715	35	72026522	64	26060252	93	51543167
7	40213222	36	34321402	65	62124406	94	27536114
8	76435525	37	25414672	66	64751452	95	44330217
9	20304547	38	23025645	67	32464531	96	42210377
10	23741070	39	35320102	68	50415046	97	45725725
11	77341334	40	15364372	69	43154545	98	12655332
12	61006774	41	71026776	70	2024051	99	00056361
13	04416500	42	33111702	71	46316402	100	63603101
14	61033145	43	25733455	72	22702375	101	54524760
15	56133407	44	11702037	73	7405230	102	36250612
16	21653557	45	52450530	74	57074340	103	15054721
17	16053661	46	76650341	75	33602603	104	75210201
18	67560701	47	56304761	76	34645760	105	52434625
19	26611116	48	34240670	77	60321064	106	37542147
20	71717365	49	17065024	78	32573272	107	33314714
21	47565235	50	04017106	79	45341513	108	55635475
22	13757116	51	75457105	80	50157574	109	46514067
23	21053031	52	41212241	81	60321151	110	06334315
24	11455601	53	61575225	82	33443272	111	45414577
25	00014641	54	77620013	83	40717344	112	00277645

<i>Exp</i>	<i>Polynomial</i>	<i>Exp</i>	<i>Polynomial</i>	<i>Exp</i>	<i>Polynomial</i>	<i>Exp</i>	<i>Polynomial</i>
138	34776723	167	32516775	196	05035614	225	32521205
139	70253733	168	72357715	197	55465007	226	25406412
140	00400121	169	46335371	198	42067746	227	41276140
141	01654040	170	11650577	199	31626006	228	20507023
142	43561661	171	37556661	200	70033762	229	56363010
143	53657516	172	70563411	201	00130707	230	25552476
144	61162061	173	05366616	202	07372757	231	01316211
145	45675356	174	65561776	203	24740673	232	65166275
233	3375756	239	45006041	245	11547715	251	26254350
234	50370273	240	23324100	246	21412114	252	23737721
235	75151373	241	20027172	247	2605245	253	56622637
236	63023251	242	57100202	248	50416260	254	13055062
237	55064102	243	72546350	249	55374545	255	00000001
238	33453706	244	32510614	250	37101473		

Table 6.1: Elements of maximal cyclic subgroup G_{255}

Step 2: Define a mapping for $a = \beta^5 \in G_{255}$, $\rho: G_{255} \cup \{0\} \rightarrow G_{255} \cup \{0\}$ by

$$\rho(3\alpha^7 + 2\alpha^6 + 5\alpha^5 + \alpha^4 + 6\alpha^3 + 5\alpha^2 + 6\alpha + 2) = 3\alpha^7 + 7\alpha^6 + \alpha^5 + \alpha^3 + 4\alpha^2 + 7\alpha + 3. \quad (6.4)$$

Step 3: Select any fixed element b from G_{255} , for instance, $b = \beta^{40} = \alpha^7 + 5\alpha^6 + 3\alpha^5 + 6\alpha^4 + 4\alpha^3 + 3\alpha^2 + 7\alpha + 2$ then apply mapping,

$$\begin{aligned} \sigma: G_{255} \cup \{0\} &\rightarrow G_{255} \cup \{0\} \\ \sigma(a) &= (a)(\beta^{40}) \text{ for all } a \in G_{255} \end{aligned} \quad (6.5)$$

Step 4: Find the elements of S-Box using the mapping $\rho \circ \sigma: G_{255} \cup \{0\} \rightarrow G_{255} \cup \{0\}$ by

$$\rho \circ \sigma(a) = (a\beta^{40})^{-1} \text{ for all } a \in G_{255}. \quad (6.6)$$

47314104	33314714	37506535	71717214	11702037	47565235	56622637	25552476
01044440	37542147	40717344	26541144	25733455	71717365	23737721	56363010
04400504	52434625	33443272	51443017	33111702	26611116	26254350	20507023
05153450	75210201	60321151	15532357	71026776	67560701	37101473	41276140
64106742	15054721	50157574	04131167	67225333	16053661	55374545	25406412
41422521	36250612	45341513	62636436	50777444	21653557	50416260	32521205
14251651	54524760	32573272	56767562	30444347	56133407	02605245	21015142
51152672	63603101	60321064	77620013	74436313	61033145	21412114	73425740
14322676	00056361	34645760	61575225	40573013	04416500	11547715	53400727
50362113	02635357	33602603	41212241	51575262	61006774	32510614	50236051
06575620	74133547	57074340	75457105	50626627	32516562	72546350	63153546
76203323	26431410	07405230	04017106	33234244	27625706	57100202	13463330
00277645	04502214	22702375	17065024	46447505	75065707	20027172	07701662
45414577	16545373	46316402	34240670	00014641	00037505	23324100	14625444
06334315	10376772	02024051	56304761	11455601	00010000	45006041	10446552
46514067	51322227	10127701	76650341	21053031	00000001	33453706	77124254
55635475	77230310	46055235	52450530	13757116	13055062	01316211	00000000

Table 6.2: Elements of S-Box using G_{255}

If we take another element b from G_{255} and apply the scheme for construction of S-Box, we get another S-Box over the Galois ring $GR(8, 8)$. We get 255 different S-Boxes corresponding to each element b of G_{255} .

6.1.2 Scheme for Nonlinear Component over the Galois Field

S-Boxes are constructed over Galois field $GF(2^8)$ is the same as S-Boxes over the Galois ring $GR(8, 8)$. By using the following method,

Step 1: Find the elements of Galois field $GF(2^8)$.

<i>Exp</i>	<i>Polynomial</i>	<i>Exp</i>	<i>Polynomial</i>	<i>Exp</i>	<i>Polynomial</i>	<i>Exp</i>	<i>Polynomial</i>
1	00000010	30	00111101	59	11011111	88	11001010
2	00000100	31	01111010	60	11011101	89	11110111
3	00001000	32	11110100	61	11011001	90	10001101
4	00010000	33	10001011	62	11010001	91	01111001
5	00100000	34	01110101	63	11000001	92	11110010
6	01000000	35	11101010	64	11100001	93	10000111
7	10000000	36	10110111	65	10100001	94	01101101
8	01100011	37	00001101	66	00100001	95	11011010
9	11000110	38	00011010	67	01000010	96	11010111
10	11101111	39	00110100	68	10000100	97	11001101
11	10111101	40	01101000	69	01101011	98	11111001
12	00011001	41	11010000	70	11010110	99	10010001
13	00110010	42	11000011	71	11001111	100	01000001
14	01100100	43	11100101	72	11111101	101	10000010
15	11001000	44	10101001	73	10011001	102	01100111
16	11110011	45	00110001	74	01010001	103	11001110
17	10000101	46	1100010	75	10100010	104	11111111
18	01101001	47	11000100	76	00100111	105	10011101
19	11010010	48	11101011	77	01001110	106	01011001
20	11000111	49	10110101	78	10011100	107	10110010
21	11101101	50	00001001	79	01011011	108	00000111
22	10111001	51	00010010	80	10110110	109	00001110
23	00010001	52	00100100	81	00001111	110	00011100
24	00100010	53	01001000	82	00011110	111	111000
25	01000100	54	10010000	83	00111100	112	01110000

<i>Exp</i>	<i>Polynomial</i>	<i>Exp</i>	<i>Polynomial</i>	<i>Exp</i>	<i>Polynomial</i>	<i>Exp</i>	<i>Polynomial</i>
205	10100101	218	1010100	231	10011111	244	00101111
206	00101001	219	10101000	232	01011101	245	01011110
207	01010010	220	00110011	233	10111010	246	10111100
208	10100100	221	01100110	234	00010111	247	00011011
209	00101011	222	11001100	235	00101110	248	00110110
210	01010110	223	11111011	236	1011100	249	01101100
211	10101100	224	10010101	237	10111000	250	11011000
212	00111011	225	01001001	238	00010011	251	11010011
213	01110110	226	10010010	239	00100110	252	11000101
214	11101100	227	01000111	240	01001100	253	11101001
215	10111011	228	10001110	241	10011000	254	10110001
216	00010101	229	1111111	242	01010011	255	00000001
217	00101010	230	11111110	243	10100110		00000000

Table 6.3: Elements of Galois field $GF(2^8)$

Step 2: Define a mapping $\rho: GF(2^8) \rightarrow GF(2^8)$ by

$$\rho(x^7 + x^5 + x^2 + 1) = (x^7 + x^5 + x^2 + 1)^{-1} = x^3 + 1 \text{ for } \alpha^{205} \in GF(2^8) \setminus \{0\}. \quad (6.10)$$

Step 3: Select an element $b = \alpha^8 = x^6 + x^5 + x + 1$ then apply mapping $\sigma: GF(2^8) \rightarrow GF(2^8)$ defined by

$$\sigma(c) = (c)(\alpha^8) \text{ for all } c \in GF(2^8). \quad (6.11)$$

Step 4: Find the elements of S-Box using the composition of two mapping

$\rho \circ \sigma: GF(2^8) \rightarrow GF(2^8)$ defined by

$$\rho \circ \sigma(c) = ((c)(\alpha^8))^{-1}, \text{ for all } c \in GF(2^8). \quad (6.12)$$

6.2 Application of Nonlinear Component in Image Encryption

In chapters 4 and 5, we encrypt the image and text by using the BCH codes and the modified AES algorithm, but the security layer was missing in that algorithm. We computed in this chapter S-Boxes using the Galois ring and the Galois field and apply those S-Boxes in data security. S-Boxes to apply in a distinctive type of cryptosystems for the security of the data is need of the day. We encrypt the image by using the following algorithm, and this algorithm has 10 rounds, each round consist of following steps,

Step 1: Construct the secret key of 128 bit by using BCH codes over the Galois ring or the Galois field.

Step 2: Divide the pixels of the image into a block of 128 bits and perform the XOR operation with a secret key.

Step 3: Construct the S-Box by using the elements of the Galois ring or the Galois field.

Step 4: Apply the S-Box to the pixels of the image after secret key addition.

Step 5: Construct maximum distance separable matrix using the BCH codes over the Galois ring or Galois field.

Step 6: Multiply the MDS matrix with pixels of the image after applying S-Box.

Step 7: Repeat Step 1 to Step 6 and construct different keys by using BCH code, different S-Boxes, different MDS matrices for each round.

Example 57 : *Suppose that we want to encrypt an image by using the above scheme, then we perform the following steps,*

Step 1: Construction of Secret keys for each round

To find the secret keys by using BCH codes of length 255 over Galois field $GF(2^8)$, find the generator polynomial for designed distance $d = 119$,

For designed distance $d = 87$,

Key 5 = 97 83 84 35 104 195 56 41 224 158 27 124 162 63 115 89.

For designed distance $d = 85$,

Key 6 = 123 198 170 165 96 163 64 9 147 101 184 77 233 179 201 224.

For designed distance $d = 63$,

Key 7 = 143 119 118 176 72 226 45 130 246 214 24 86 99 161 193 88.

For designed distance $d = 60$,

Key 8 = 193 198 10 119 189 248 86 75 235 73 108 15 56 254 238 81.

For designed distance $d = 58$,

Key 9 = 29 140 71 26 253 86 30 42 146 133 230 100 83 162 135 109.

For designed distance $d = 55$,

Key 10 = 237 196 133 181 159 220 90 131 200 124 234 29 34 135 112.

Step 2: Construction of S-Boxes over the Galois field $GF(2^8)$

Compute S-Boxes for each round by using mapping defined in equation 6.9.

0	E9	FA	7D	B0	F8	2C	16	B	8B	CB	EB	FB	F3	F7	F5
F4	7A	3D	90	48	24	12	9	8A	45	AC	56	2B	9B	C3	EF
F9	F2	79	B2	59	A2	51	A6	53	A7	DD	E0	70	38	1C	E
7	8D	C8	64	32	19	82	41	AE	57	A5	DC	6E	37	95	C4
62	31	96	4B	AB	DB	E3	FF	F1	F6	7B	B3	D7	E5	FC	7E
3F	91	C6	63	BF	D1	E6	73	B7	D5	E4	72	39	92	49	AA
55	A4	52	29	9A	4D	A8	54	2A	15	84	42	21	9E	4F	A9
DA	6D	B8	5C	2E	17	85	CC	66	33	97	C5	EC	76	3B	93
C7	ED	F8	7C	3E	1F	81	CE	67	BD	D0	68	34	1A	D	88
44	22	11	86	43	AF	D9	E2	71	B6	5B	A3	DF	E1	FE	7F
B1	D6	6B	BB	D3	E7	FD	F0	78	3C	1E	F	89	CA	65	BC
5E	2F	99	C2	61	BE	5F	A1	DE	6F	B9	D2	69	BA	5D	A0
50	28	14	A	5	8C	46	23	9F	C1	EE	77	B5	D4	6A	35
94	4A	25	9C	4E	27	9D	C0	60	30	18	C	6	3	8F	C9
EA	75	B4	5A	2D	98	4C	26	13	87	CD	E8	74	3A	1D	80
40	20	10	8	4	2	1	8E	47	AD	D8	6C	36	1B	83	CF

Table 6.6: Elements of S-Box 2 over $GF(2^8)$

0	7D	B0	58	2C	16	B	8B	CB	EB	FB	F3	F7	F5	F4	7A
3D	90	48	24	12	9	8A	45	AC	56	2B	9B	C3	EF	F9	F2
79	B2	59	A2	51	A6	53	A7	DD	E0	70	38	1C	E	7	8D
C8	64	32	19	82	41	AE	57	A5	DC	6E	37	95	C4	62	31
96	4B	AB	DB	E3	FF	F1	F6	7B	B3	D7	E5	FC	7E	EF	91
C6	63	BF	D1	E6	73	B7	D5	E4	72	39	92	49	AA	55	A4
52	29	9A	4D	A8	54	2A	15	84	42	21	9E	4F	A9	DA	6D
B8	5C	2E	17	85	CC	66	33	97	C5	EC	76	3B	93	C7	ED
F8	7C	3E	1F	81	CE	67	BD	D0	68	34	1A	D	88	44	22
11	86	43	AF	D9	E2	71	B6	5B	A3	DF	E1	FE	7F	B1	D6
6B	BB	D3	E7	FD	F0	78	3C	1E	F	89	CA	65	BC	5E	2F
99	C2	61	BE	5F	A1	DE	6F	B9	D2	69	BA	5D	A0	50	28
14	A	5	8C	46	23	9F	C1	EE	77	B5	D4	6A	35	94	4A
25	9C	4E	27	9D	C0	60	30	18	C	6	3	8F	C9	EA	75
B4	5A	2D	98	4C	26	13	87	CD	E8	74	3A	1D	80	40	20
10	8	4	2	1	8E	47	AD	D8	6C	36	1B	83	CF	E9	FA

Table 6.8: Elements of S-Box 4 over $GF(2^8)$

0	CB	EB	FB	F3	F7	F5	F4	7A	3D	90	48	24	12	9	8A
45	AC	56	2B	9B	C3	EF	F9	F2	79	B2	59	A2	51	A6	53
A7	DD	E0	70	38	1C	E	7	8D	C8	64	32	19	82	41	AE
57	A5	DC	6E	37	95	C4	62	31	96	4B	AB	DB	E3	FF	F1
F6	7B	B3	D7	E5	FC	7E	3F	91	C6	63	BF	D1	E6	73	B7
D5	E4	72	39	92	49	AA	55	A4	52	29	9A	4D	A8	54	2A
15	84	42	21	9E	4F	A9	DA	6D	B8	5C	2E	17	85	CC	66
33	97	65	EC	76	3B	93	C7	ED	F8	7C	3E	1F	81	CE	67
BD	D0	68	34	1A	D	88	44	22	11	86	43	AF	D9	E2	71
B6	5B	A3	DF	E1	FE	7F	B1	D6	6B	BB	D3	E7	FD	F0	78
3C	1E	F	89	CA	65	BC	5E	2F	99	C2	61	BE	5F	A1	DE
6F	B9	B2	69	BA	FD	A0	50	28	14	A	5	8C	46	23	9F
C1	EE	77	B5	D4	6A	35	94	4A	25	9C	4E	27	9D	C0	60
30	18	C	6	3	8F	C9	EA	75	B4	5A	2D	98	4C	26	13
87	CD	E8	74	3A	1D	80	40	20	10	8	4	2	1	8E	47
AD	D8	6C	36	1B	83	CF	E9	FA	7D	B0	58	2C	16	B	8B

Table 6.10: Elements of S-Box 6 over $GF(2^8)$

0	90	48	24	12	9	8A	45	AC	56	2B	9B	C3	EF	F9	F2
79	B2	59	A2	51	A6	53	A7	DD	E0	70	38	1C	E	7	8D
C8	64	32	19	82	41	AE	57	A5	DC	6E	37	95	C4	62	31
96	4B	AB	DB	E3	FF	F1	F6	7B	B3	D7	E5	FC	7E	3F	91
C6	63	BF	D1	E6	73	B7	D5	E4	72	39	92	49	AA	55	A4
52	29	9A	4D	A8	54	2A	15	84	42	21	9E	4F	A9	DA	6D
B8	5C	2E	17	85	CC	66	33	97	C5	EC	76	3B	93	C7	ED
F8	7C	3E	1F	81	CE	67	BD	D0	68	34	1A	D	88	44	22
11	86	43	AF	D9	E2	71	B6	5B	A3	DF	E1	FE	7F	B1	D6
6B	BB	D3	E7	FD	F0	78	3C	1E	F	89	CA	65	BC	5E	2F
99	C2	61	BE	5F	A1	DE	6F	B9	D2	69	BA	5D	A0	50	28
14	A	5	8C	46	23	9F	C1	EE	77	B5	D4	6A	35	94	4A
25	9C	4E	27	9D	C0	60	30	18	C	6	3	8F	C9	EA	75
B4	5A	2D	98	4C	26	13	87	CD	E8	74	3A	1D	80	40	20
10	8	4	2	1	8E	47	AD	D8	6C	36	1B	83	CF	E9	FA
7D	B0	58	2C	16	B	8B	CB	EB	FB	F3	F7	F5	F4	7A	3D

Table 6.12: Elements of S-Box 8 over $GF(2^8)$

0	51	A6	53	A7	DD	E0	70	38	1C	E	7	8D	C8	64	32
19	82	41	AE	57	A5	DC	6E	37	95	C4	62	31	96	4B	AB
DB	E3	FF	F1	F6	7B	B3	D7	E5	FC	7E	3F	91	C6	63	BF
D1	E6	73	B7	D5	E4	72	39	92	49	AA	55	A4	52	29	9A
4D	A8	54	2A	15	84	42	21	9E	4F	A9	DA	6D	B8	5C	2E
17	85	CC	66	33	97	C5	EC	76	3B	93	C7	ED	F8	7C	3E
1F	81	CE	67	BD	D0	68	34	1A	D	88	44	22	11	86	43
AF	D9	E2	71	B6	5B	A3	DF	E1	FE	7F	B1	D6	6B	BB	D3
E7	FD	F0	78	3C	1E	F	89	CA	65	BC	5E	2F	99	C2	61
BE	5F	A1	DE	6F	B9	D2	69	BA	5D	A0	50	28	14	A	5
8C	46	23	9F	C1	EE	77	B5	D4	6A	35	94	4A	25	9C	4E
27	9D	C0	60	30	18	C	6	3	8F	C9	EA	75	B4	5A	2D
98	4C	26	13	87	CD	E8	74	3A	1D	80	40	20	10	8	4
2	1	8E	47	AD	D8	6C	36	1B	83	CF	E9	FA	7D	B0	58
2C	16	B	8B	CB	EB	FB	F3	F7	F5	F4	7A	3D	90	48	24
12	9	8A	45	AC	56	2B	9B	C3	EF	F9	F2	79	B2	59	A2

Table 6.14: Elements of S-Box 10 over $GF(2^8)$

Step 3: Construction of different MDS matrices for each round

The scheme of construction of maximum distance separable matrices is already explained in chapter 5, section 5.2. By using the BCH codes over the Galois field, different MDS matrices for each round are calculated as,

$$A_1 = \begin{pmatrix} E9 & 44 & 49 & 39 \\ 15 & 61 & 09 & A1 \\ F4 & 70 & 04 & 74 \\ A4 & CF & 31 & 23 \end{pmatrix}, A_2 = \begin{pmatrix} DD & 68 & B6 & 59 \\ 6E & B4 & 5B & 2C \\ EA & 32 & 9B & CF \\ A8 & 71 & FB & BE \end{pmatrix}, A_3 = \begin{pmatrix} D4 & 75 & 75 & 91 \\ BE & 4F & CF & 59 \\ 5F & 27 & E7 & AC \\ FB & E6 & 86 & 47 \end{pmatrix}$$



Fig. 6.3: Plain Image

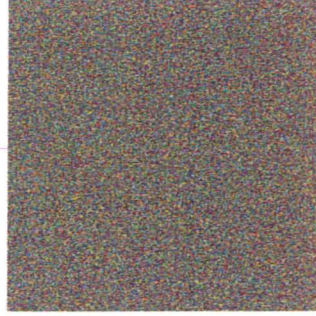


Fig. 6.4: Cipher Image

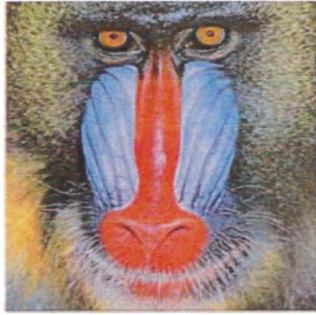


Fig. 6.5: Plain Image

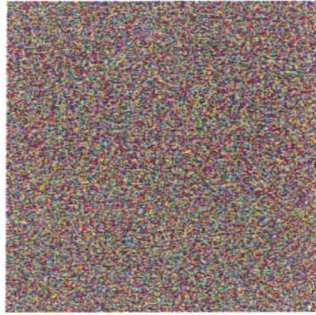


Fig. 6.6: Cipher Image

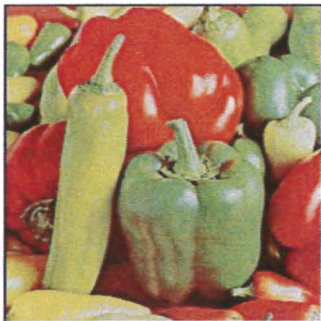


Fig. 6.7: Plain Image



Fig. 6.8: Cipher Image

The Statistical analysis of encrypted image of Lena,mandrill, pepper and deblur through blue channel are shown in Table 6.17,

Image	Contrast	Correlation	Energy	Homogeneity	Entropy
Deblur	10.5257	-0.0026	0.0156	0.3890	7.9969
Lena	10.5075	0.0000199	0.0156	0.3899	7.9973
Mandrill	10.5321	-0.0043	0.0156	0.3893	7.9971
Pepper	10.5075	0.0000199	0.0156	0.3899	7.9973

Table 6.17: Statistical analysis through blue channel

6.2.3 Histogram Analysis

The uniformity of the image's histogram of an encrypted image is the finest aspect for assessing image encryption systems' security. We analyze here the color Deblur, Lena, Mandrill, and Pepper image of dimension 256×256 , which have different contents, and its histogram is considered. Histogram of the encrypted image under the proposed algorithm is likewise identical and different from plain image, which makes statistical attacks hard. We have drawn three-dimensional 3D histograms for plain and encrypted images to study the uniformity in encrypted image. The histogram trickle holds the data distribution of pixel respects in a picture. A flawless encrypted picture should have a uniform histogram spreading to preserve the rival from separating any supportive data from the unstable histogram. The 3D histograms of the original and encrypted image of Deblur, Lena, Mandrill, and Pepper are shown in Fig. 6.9 to 6.16.

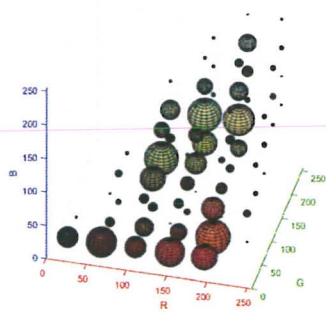


Fig. 6.15: Pepper original

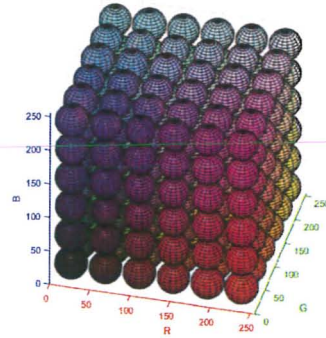


Fig. 6.16: Pepper encrypted

6.2.4 Analyses Discussion

Initially, statistical analyses are performed. Information entropy and correlation determine the security strength of the algorithm. Our entropy is approaching 8, and correlation is very closed to zero. These outcomes indicate that the encryption scheme has highly disordered, and the relationship between pixels is handsomely break. On the other hand, the quality of encryption is determined through contrast, energy, and homogeneity. The outcomes from Tables 6.15, 6.16, and 6.17 indicates that the encryption scheme is highly secure and has outstanding encryption quality.

The histogram analyses also determine security strength. From figures, Fig. 6.10, 6.12, 6.14, 6.16, the encrypted image has a uniform distribution, which provides justification that the anticipated scheme has outstanding resistance capability against histogram attacks.

Chapter 7

Conclusion

In this thesis, we have presented two types of BCH codes one is over Galois ring, and the second is over the Galois field. The relation between these two types of codes is developed computationally so that data can be transmitted through any of the BCH codes over the Galois ring or Galois field. The selection of code is based on the choice of better code rate or error correction capability of the chosen code or consistency in transmission. Since the number of code words in the BCH code over the Galois ring is greater than the BCH code over the Galois field, therefore, we can transfer maximum information per unit time in BCH codes in the Galois ring. One can transmit data easily from the Galois ring to the Galois field by using the proposed computational approach. We can choose better error correction capabilities of BCH codes computationally by changing the input algorithm's designed distance. A computational method is designed to construct the BCH codes over the Galois ring for encoding our messages. We have constructed a computationally maximal cyclic subgroup of any order as desired, which helps us to construct S-box over Galois ring in the field of Cryptography and decoding of BCH codes. We have also found the dimension of the primitive BCH code computationally for any fixed length with the degree of the generator polynomial. The problem of the dimensions of primitive BCH codes over the Galois field is resolved. This novel approach provides us the generator polynomial for BCH codes of each length over the Galois ring and Galois field. If the data can be encoded either by the BCH codes over the Galois ring or BCH codes over the Galois field then transmitted data can also be decoded using the computational approach of the Berlekamp Massey algorithm for the Galois field-based BCH code. The luxury of encoding

References

- [1] Blake, Ian F. "Codes over certain rings." *Information and Control* 20, no. 4 (1972): 396-404.
- [2] Blake, Ian F. "Codes over integer residue rings." *Information and Control* 29, no. 4 (1975): 295-300.
- [3] Bose, Raj Chandra, and Dwijendra K. Ray-Chaudhuri. "On a class of error correcting binary group codes." *Information and control* 3, no. 1 (1960): 68-79.
- [4] Ding, Cunsheng, Cuiling Fan, and Zhengchun Zhou. "The dimension and minimum distance of two classes of primitive BCH codes." *Finite Fields and Their Applications* 45 (2017): 237-263.
- [5] De Andrade, Antonio Aparecido, and Reginaldo Palazzo Jr. "Construction and decoding of BCH codes over finite commutative rings." *Linear Algebra and Its Applications* 286, no. 1-3 (1999): 69-85.
- [6] Daemen, Joan, and Vincent Rijmen. "AES proposal: Rijndael." (1999).
- [7] Forney, George. "On decoding BCH codes." *IEEE Transactions on information theory* 11, no. 4 (1965): 549-557.
- [8] Gamido, Heidilyn V., Ariel M. Sison, and Ruji P. Medina. "Implementation of Modified AES as Image Encryption Scheme." *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)* 6, no. 3 (2018): 301-308.
- [9] Golay, Marcel JE. "Notes on digital coding." *Proc. IEEE* 37 (1949): 657.

- [21] Preishuber, Mario, Thomas Hütter, Stefan Katzenbeisser, and Andreas Uhl. "Depreciating motivation and empirical security analysis of chaos-based image and video encryption." *IEEE Transactions on Information Forensics and Security* 13, no. 9 (2018): 2137-2150.
- [22] Patel, Komal D., and Sonal Belani. "Image encryption using different techniques: A review." *International Journal of Emerging Technology and Advanced Engineering* 1, no. 1 (2011): 30-34.
- [23] Prange, Eugene. *Cyclic Error-Correcting codes in two symbols*. Air force Cambridge research center, 1957.
- [24] Prange, Eugene. *The Use of Coset Equivalence in the Analysis and Decoding of Group Codes*. No. AFCRC-TR-59-164. Air Force Cambridge Research Labs Hanscom AFB MA, 1959.
- [25] Peterson, Wesley. "Encoding and error-correction procedures for the Bose-Chaudhuri codes." *IRE Transactions on Information Theory* 6, no. 4 (1960): 459-470.
- [26] Reed, Irving S. *A class of multiple-error-correcting codes and the decoding scheme*. No. TR-44. Massachusetts Inst. of Tech. Lexington Lincoln Lab., 1953.
- [27] Rijmen, Vincent, Joan Daemen, Bart Preneel, Antoon Bosselaers, and Erik De Win. "The cipher SHARK." In *International Workshop on Fast Software Encryption*, pp. 99-111. Springer, Berlin, Heidelberg, 1996.
- [28] Subramanyan, B., Vivek M. Chhabria, and TG Sankar Babu. "Image encryption based on AES key expansion." In *2011 Second International Conference on Emerging Applications of Information Technology*, pp. 217-220. IEEE, 2011.
- [29] Schneier, Bruce, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. "Twofish: A 128-bit block cipher." *NIST AES Proposal* 15, no. 1 (1998): 23-91.
- [30] Schneier, Bruce, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson, Tadayoshi Kohno, and Mike Stay. "The Twofish team's final comments on AES Selection." *AES round 2*, no. 1 (2000): 1-13.

Appendix

This section consist of computer programs which are very helpful to understand this thesis.

```
{  
  
public partial class frmEncoding : Form  
{  
    string alpha = "$\alpha$";  
    string beta = "$\beta$";  
    Color _coeffColor = Color.Green;  
    Color _variableColor = Color.Blue;  
    Color _degreeColor = Color.Red;  
    Color _modulusColor = Color.Magenta;  
    int _q = 0;  
    int _n = 0;  
    int _d = 0;  
    int _p = 0;  
    int _blinkerCount = 1;  
    Polynomial currentPolynomial = new Polynomial();  
    int startingAlphaDegree = 4;  
    int currentAlphaDegree = 4;  
    List<Polynomial> alphaPolynomials = new List<Polynomial>();  
  
    List<Polynomial> betaPolynomials = new List<Polynomial>();  
    List<Polynomial> MiPolynomials = new List<Polynomial>();  
    List<Polynomial> DistinctMiPolynomialsModeP = new List<Polynomial>();  
    List<Polynomial> DistinctMiPolynomials = new List<Polynomial>();  
    Polynomial GeneratorPolynomialRing = new Polynomial();  
    Polynomial GeneratorPolynomialField = new Polynomial();  
    public frmEncoding()
```

```

private void btnAdd_Click(object sender, EventArgs e)
{
    _q = int.Parse(txtQ.Text.Trim());
    _n = int.Parse(txtN.Text.Trim());
    _d = int.Parse(txtD.Text.Trim());
    _p = int.Parse(txtP.Text.Trim());
    startingAlphaDegree = int.Parse(txtAlphaDegree.Text);
    btnAdd.Enabled = txtD.Enabled = txtP.Enabled = txtQ.Enabled = txtN.Enabled =
false;
    btnSolve.Enabled = true;
    List<double> alphaEq = new List<double>();
    for (int alphaPoly = 1; alphaPoly < startingAlphaDegree; alphaPoly++)
    {
        alphaEq.Clear();
        for (int poly = 0; poly < alphaPoly; poly++)
            alphaEq.Add(0);
        alphaEq.Add(1);
        alphaPolynomials.Add(new Polynomial(alphaEq.ToArray<double>()));
    }
    // for last polynomial and adding it to rTxtExpression
    alphaEq[alphaEq.Count - 1] = 0;
    alphaEq.Add(1);
    //for (int i = 0; i < startingAlphaDegree; i++)
    // alphaEq.Add(0);
    //alphaEq.Add(1);
    AddToRichTextBoxExpression(rTxtPolynomialExpression, alphaEq.ToArray(),
false, "$\alpha$");
    rTxtPolynomialExpression.AppendText(" = ");
    currentPolynomial.AddA(new Polynomial(txtPolynomial.Text.Trim()));
}

```

```

//mvNextPoly.AddTerm("d");
//mvNextPoly.SetIndeterminateValue("x", 4.0);
//MessageBox.Show(mvPoly.ToString());
//MessageBox.Show(mvNextPoly.ToString());
////Polynomial mulPolynomial = mvPoly. * mvNextPoly;
////MessageBox.Show(mulPolynomial.ToString());
/*****/
#endregion
currentAlphaDegree = startingAlphaDegree;

bool findAlphaFlag = true;
do
{
findAlphaFlag = MultiplyEquationWithAlpha(currentPolynomial);
if (rBtnFindAlpha.Checked==true && chkAlpha.Checked == true &&
findAlphaFlag == false)
{
break;
}
} while (PolynomialIsConstant(currentPolynomial) == false);
int initialPowerOfAlphaForBeta;
// condition for finding only alpha
if (rBtnFindAlpha.Checked == false)
{
if (chkBeta.Checked == true)
{
initialPowerOfAlphaForBeta = (alphaPolynomials.Count) / _n;
for (int betaPower = 1; betaPower <= _n; betaPower++)
{
Polynomial pForBeta = new Polynomial(alphaPolynomials

```

```

rTxtGeneratorPolynomial.AppendText("g(x) = ");
rTxtGeneratorPolynomialWithModeP.AppendText("g(x) = ");
AddToRichTextBoxExpression(rTxtGeneratorPolynomial, GeneratorPolynomialRing.ToArray(),
false, "X");
AddToRichTextBoxExpression(rTxtGeneratorPolynomialWithModeP, GeneratorPolynomialField.1
false, "X");
timerBlinker.Enabled = true;
}
}
else // finding alpha only
{
if (alphaPolynomials.Count % _n == 0)
{
MessageBox.Show("Found polynomial");
}
else
{
MessageBox.Show("Polynomial does not qualify");
}
}
btnTestEncodeDecod.Enabled = true;

int dummy = 0;
if (betaPolynomials.Count > 0)
{
MCSG result = MCSG.GetMcsg;
result.BetaPolynomials = betaPolynomials;
result.Q = _q;
result.N = _n;
result.P = _p;

```

```

    }
    else
    {
        foreach (Polynomial p in pList)
        {
            Poly = PolynomialModX(Poly.Mul(p),modeValue);
        }
    }
    return Poly;
}

private void AddPolynomialListToRichTextBox(List<Polynomial> listP)
{
    int i = 1;
    foreach (Polynomial p in listP)
    {
        rTxtMiModeP.AppendText("M(" + (i++).ToString() + ") = ");
        AddToRichTextBoxExpression(rTxtMiModeP, p.ToArray(), true, "x");
    }
}

private List<Polynomial> CalculateAllMiModeP(List<Polynomial> allMi)
{
    List<Polynomial> MiModeP = new List<Polynomial>();
    foreach (Polynomial p in allMi)
    {
        MiModeP.Add(PolynomialModX(p,_p));
    }

    return MiModeP;
}

```



```

AddToRichTextBoxExpression(rTxtMx,tempMkPolynomial.ToArray(),true,"x");
MiPolynomials.Add(tempMkPolynomial);
}
}
private Polynomial CalculateMk(List<int> MkBetaPowers, List<Polynomial> bPolynomials
{
int rows = MkBetaPowers.Count + 1; // assumption:
power of x would not increase than the number of polynomials in a set
int cols = betaPolynomials.Count + 1;
int totalPolynomialSets = rows-1; // e.g. (x-B)(x-B^2)(x-B^4)(x-B^8)
have 4 polynomial sets

////////// creating matrix for result //////////
int[][] multivariatePolynomialResult = Create2DIntArray(rows, cols);
//int[][] multivariatePolynomialResult = new int[rows][];
//for (int i = 0; i < rows; i++)
//{
// multivariatePolynomialResult[i] = new int[cols];
//}

//////////

////////// Creating matrices for all polynomial sets in Mk //////////
int[][][] multivariatePolynomials = new int[totalPolynomialSets][][];
for (int i = 0; i < totalPolynomialSets; i++)
{
multivariatePolynomials[i] = Create2DIntArray(rows, cols);
}
//////////

List<Polynomial> Mi = new List<Polynomial>();

```

```

}
}
}
indexesNoZeroValues.Add(tempSetIndexes);
}
CopyArray(multivariatePolynomials[0], multivariatePolynomialResult);
for (int setNo = 1 ; setNo < indexesNoZeroValues.Count ; setNo++)
{
foreach(KeyValuePair<int,int> cordIndexes in indexesNoZeroValues[setNo])
{
for(int row = 0 ; row < multivariatePolynomialResult.Length ; row++)
for(int col = 0 ; col < multivariatePolynomialResult[0].Length ; col++)
{
if(multivariatePolynomialResult[row][col] != 0 )
{
int newRowIndex;
int newColIndex;
newRowIndex = cordIndexes.Key + row;
// taken mode with n to reduce power of Beta
newColIndex = (cordIndexes.Value + col) % _n;

int newCoefficient;
newCoefficient = multivariatePolynomials[setNo][cordIndexes.Key]
[cordIndexes.Value] * multivariatePolynomialResult[row][col];
//tempResult[newRowIndex][newColIndex] = multivariatePolynomialResult[newRowIndex]
[newColIndex] + newCoefficient;
tempResult[newRowIndex][newColIndex] = tempResult[newRowIndex]
[newColIndex] + newCoefficient;
}
}
}
}

```

```

// this will happend only in the form when taking common, overall constant of
the
polynomial will be added later
// x = 0 and beta = 0 is the overall polynomial constant
if (x > 0)
{
constantPoly[0] = multivariatePolynomialResult[x][0];
tempPolynomial.AddA(constantPoly);
}

for (int coeff = 0; coeff <= tempPolynomial.MaxIndex; coeff++)
tempPolynomial[coeff] = (tempPolynomial[coeff] % _q + _q) % _q;
if (tempPolynomial.MaxIndex == -1 && x == multivariatePolynomialResult.Length
- 1)
Mk.AddA(new Polynomial("x^" + x.ToString()));
else
Mk.AddA(new Polynomial(tempPolynomial.Mul(new Polynomial
("x^" + x.ToString())).ToArray()));
}
Mk[0] += (multivariatePolynomialResult[0][0] % _q + _q) % _q;
return Mk;
}

private void InitializeMultivariatePolynomials(int[] [] []
MVPolynomials, List<int> betaPowers)
{
for (int MVPolyIndex = 0; MVPolyIndex < MVPolynomials.Length ; MVPolyIndex++)
{
// represents X^1 in matrix
MVPolynomials[MVPolyIndex][1][0] = 1;

```

```

}
return a;
}
#endregion

private void ShowPolynomialMatrix(int[] [] polynomilaMatrix)
{
string msg = "";
for (int t = 0; t < polynomilaMatrix.Length; t++)
{
for (int x = 0; x < polynomilaMatrix[0].Length; x++)
msg += polynomilaMatrix[t][x].ToString() + " ";
msg += "\n";
}
MessageBox.Show(msg);
}
#region " Incorrect Implimentation using first substitution they multiplication
"

private void CalculateAllMiPolynomials(List<List<int>> MiAndBetaPowers)
{
List<int> MkBetaPowers = MiAndBetaPowers[0];
Polynomial Mk;
for (int k = 0; k < MiAndBetaPowers.Count; k++)
{
Mk = new Polynomial(CalculateMkPolynomial(MiAndBetaPowers[k]).ToArray());
MiPolynomials.Add(new Polynomial(Mk));
rTxtMx.AppendText("M" + (k+1).ToString() + "(x) = ");
AddToRichTextBoxExpression(rTxtMx, Mk.ToArray(), true,"x");
rTxtMx.AppendText("\n\n");
}
}

```

```

//power = (i * p * k) % n;
//k++;
if (power > (alphaPolynomials.Count - 1))
{
    MessageBox.Show("$\beta$" + power.ToString() + "
does not exist in polynomial list... \nSomething has gone wrong");
    throw new Exception("$\beta$" + power.ToString() + "
does not exist in polynomial list... \nSomething has gone wrong");
}
if (tempBetaPowers.Contains(power))
break;
else
tempBetaPowers.Add(power);
} while (true);
tempBetaPowers.Sort();
return tempBetaPowers;
}
private bool PolynomialIsConstant(Polynomial p)
{
    for (int coeffIndex = 1; coeffIndex <= p.MaxIndex; coeffIndex++)
    {
        if (p[coeffIndex] != 0)
            return false;
    }
    // special condition that the polynomial should not only be a constant but also
    '1'
    if ( p[0] == 1 )
        return true;
    return false;
}

```

```

{
if (temp == " - ")
temp += (-polyCoeff[index]).ToString();
else
temp += polyCoeff[index].ToString();
}
}

previousIndex = rtxt.Text.Length - 1;
rtxt.AppendText(temp);
startIndex = rtxt.Text.IndexOf(temp, previousIndex,
StringComparison.InvariantCultureIgnoreCase);
rtxt.Select(startIndex, temp.Length);
rtxt.SelectionColor = _coeffColor;
// last constant in polynomial, therefore no need to print variable
if (index != 0)
{
temp = variable;
previousIndex = rtxt.Text.Length - 1;
rtxt.AppendText(temp);
startIndex = rtxt.Text.IndexOf(temp, previousIndex,
StringComparison.InvariantCultureIgnoreCase);
rtxt.Select(startIndex, temp.Length);
rtxt.SelectionColor = _variableColor;
}

// no need to print degree when it is 1
if (index != 1 && index != 0)
{
temp = index.ToString();

```

```

rtxt.AppendText(" (+ binary + " )");
if ( newLine)
rtxt.AppendText("\n");

}

private bool MultiplyEquationWithAlpha(Polynomial current)
{
Polynomial p = new Polynomial("x");
current.MulA(p);
rTxtAlpha.AppendText("$\alpha$" + (++currentAlphaDegree).ToString() + " = ");
//AddToRichTextBoxExpression(rTxtAlpha, current.ToArray(), true);

SubstitutecurrentPolynomial(current);
if (alphaPolynomials.Count > _n)
return false;
return true;
}

private void SubstitutecurrentPolynomial(Polynomial current)
{
double[] coefficients = current.ToArray();
if (coefficients.Length >= startingAlphaDegree+1 && coefficients[startingAlphaDegree]
!= 0)
{
string tempCoeff = "";
if (currentPolynomial[startingAlphaDegree] < 0)
tempCoeff += "-" + current[startingAlphaDegree].ToString();
else
tempCoeff += current[startingAlphaDegree].ToString();
}
}

```

```

}

private void rBtnFindAlpha_CheckedChanged(object sender, EventArgs e)
{
    grpBoxCompute.Enabled = rBtnCompute.Checked;
}

private void chkBoxes_CheckedChanged(object sender, EventArgs e)
{
    CheckBox chkBox = (CheckBox)sender;
    //MessageBox.Show(chkBox.Checked.ToString());
    if (chkBox.Name.Equals("chkAlpha"))
    {
        if (chkBox.Checked == false)
        {
            chkBeta.Checked = false;
            chkMinimal.Checked = false;
            chkGenerator.Checked = false;
        }
        chkBox.Checked = true;
    }
    else if (chkBox.Name.Equals("chkBeta"))
    {
        if (chkBeta.Checked == false)
        {
            chkMinimal.Checked = false;
            chkGenerator.Checked = false;
        }
        else
        {
            chkAlpha.Checked = true;
        }
    }
}

```