TECHNICAL UNIVERSITY OF CRETE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Reed-Solomon Burst Error Decoding

by

Ioannis Grypiotis

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DIPLOMA IN

ELECTRICAL AND COMPUTER ENGINEERING

June 2018

THESIS COMMITTEE

Associate Professor George N. Karystinos *Thesis Supervisor*
Associate Professor Aggelos Bletsas
Assistant Professor Daphne Manoussaki

Abstract

Reed-Solomon (RS) codes are some of the oldest error detection and correction codes. However, they are widely used today in various applications, including communications and storage systems. In this work, we study their structure and their conventional encoding and decoding. Furthermore, we implement a novel burst error correction algorithm for RS codes. As a prerequisite for this work, we first present a short introduction to Galois Fields and the Bose-Chaudhuri-Hocquenghem (BCH) codes, a family of codes which are closely related to RS codes.

Acknowledgements

I would like to express my gratitude to my supervisor, Associate Professor George N. Karystinos, who gave me the opportunity to study by his side. For his encouragement, his dedication, his involvement, and guidance throughout every step of this thesis. I would also like to thank the rest of my thesis committee, Associate Professor Aggelos Bletsas and Assistant Professor Daphne Manoussaki.

I am also very grateful to my friends for always being by my side in the good times and in the bad times.

Last but not least, I would like to thank my family, my parents Thanasis and Marijke, my sister Lenna, and my brother Daniel for their endless encouragement and love.

Table of Contents

Chapter 1

Introduction to Galois fields

1.1 Integer Properties

In this chapter, we lay down our mathematical tools used through this thesis. More specifically, we start off with some basic integer properties, definitions, and theorems, followed by group, ring, and field definitions, and then we shortly introduce finite fields. We discuss their structure, basic properties, and some theorems which will be proven useful later on.

Theorem 1.1. ***(Division algorithm)*** *For any integers a and b with $a > 0$, there exists unique integers q and r such that*

$$a = qb + r$$

where $0 \leq r < b$. The number q is said to be the quotient and the number r the remainder.

By $a \pmod b$ we denote the operation of taking the remainder after dividing a by b. Now, if $a \pmod b = 0$, we say that b divides a or that b is a divisor of a.

Definition 1.1. *An integer greater than 1 is said to be prime, if it can only be divided by 1 and itself.*

By $\gcd(a, b)$ we denote the greatest common divisor of a and b, if $\gcd(a, b) = 1$, then a and b are said to be relatively prime.

Definition 1.2. *The **Euler totient function** $\phi(n)$ is the number of positive integers less than n that are relatively prime to n.*

It can be shown that [1]

$$\phi(n) = n \prod_{p|n} \frac{p-1}{p} \tag{1.1}$$

where the product is taken over all primes p dividing n. A few examples of the function $\phi(n)$ follow.

i. $\phi(4) = 4 \cdot \frac{2-1}{2} = 2$ (the numbers 1 and 3 are relatively prime to 4).

ii. $\phi(63) = \phi(3 \cdot 3 \cdot 7) = 63 \cdot \frac{2}{3} \cdot \frac{6}{7} = 36$.

iii. $\phi(64) = \phi(2^6) = 64 \cdot \frac{1}{2} = 32$.

iv. $\phi(154) = \phi(2 \cdot 7 \cdot 11) = 154 \cdot \frac{1}{2} \cdot \frac{6}{7} \cdot \frac{10}{11} = 60$.

Theorem 1.2. ***Fermat's little theorem*** *[1]: If p is a prime and if a is an integer such that $\gcd(a,p)=1$, then p divides $a^{p-1}-1$. That is if $\gcd(a,p)=1$ with p prime, then*

$$a^{p-1}-1 \equiv 0 \pmod{p}.$$

Some examples, for the prime number $p=3$, are shown below.

i. For $a=2$, it holds that $\gcd(2,3)=1$ and therefore $p=3$ divides $a^{p-1}-1=2^2-1=3$.

ii. For $a=4$, it holds that $\gcd(4,3)=1$ and therefore $p=3$ divides $a^{p-1}-1=4^2-1=15$.

iii. For $a=10$, it holds that $\gcd(10,3)=1$ and therefore $p=3$ divides $a^{p-1}-1=10^2-1=99$.

Theorem 1.3. ***Euler's generalization of Fermat's little theorem*** *[1]: If n and a are integers such that $\gcd(a,n)=1$, then:*

$$a^{\phi(n)}-1 \equiv 0 \pmod{n}$$

where ϕ is the Euler totient function.

If n is a prime then $\phi(n)=n-1$ and we get Fermat's little theorem.

1.2 Groups, Rings, and Fields

Definition 1.3. *A group $<\mathbb{G},*>$ is a set $\mathbb{G}$ of objects with a binary operation $*$ such that:*

1. *$\mathbb{G}$ is closed under the operation $*$, i.e., $\forall a,b \in \mathbb{G}$ it holds that $a*b \in \mathbb{G}$.*
2. *The operator $*$ is associative, i.e., $\forall a,b,c \in \mathbb{G}$ it holds that $(a*b)*c = a*(b*c)$.*
3. *There exists an element $e \in \mathbb{G}$ such that $a*e=e*a=a$, $\forall a \in \mathbb{G}$. The element e is called the* ***identity element****.*
4. *For every element a in $\mathbb{G}$ there exist an element b in $\mathbb{G}$ such that $a*b=e$. The element b is said to be the* ***inverse*** *of a.*

A group $<\mathbb{G},*>$ is said to be **commutative** if $a*b=b*a$, $\forall a,b \in \mathbb{G}$.

As an example of a group, let $<\mathbb{Z}_3,\oplus>$ denote the addition of the numbers $\{0,1,2\}$ modulo 3. $<\mathbb{Z}_3,\oplus>$ satisfies the definition of a group, as it can be seen in the table below.

$\oplus$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Clearly the set $\mathbb{Z}_3$ is closed under the operation $\oplus$, the element 0 is the identity element. The element 0 appears in every row and column of the table which means that there exist an inverse for every element. The associative property can also be confirmed, thus $< \mathbb{Z}_3, \oplus >$ forms a group. Furthermore, the symmetry of the table shows that $< \mathbb{Z}_3, \oplus >$ is an commutative group.

Similarly, let $< \mathbb{Z}_2, \oplus >$ denote the addition of the numbers $\{0, 1\}$ modulo 2 and let $< \mathbb{Z}_4, \oplus >$ denote the addition of the numbers $\{0, 1, 2, 3\}$ modulo 4. The respective addition tables follow.

$\oplus$	0	1
0	0	1
1	1	0

$\oplus$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	2	0	1	2

Once again, it can be confirmed that $< \mathbb{Z}_2, \oplus >$ and $< \mathbb{Z}_4, \oplus >$ form commutative groups. In fact, we can generalize the above for any set $\mathbb{Z}_n$ with $n \geq 2$, as described in the following theorem whose proof is given in the Appendix.

Theorem 1.4. *Let $\mathbb{Z}_n$, with $n \geq 2$, be the set $\{0, 1, 2, ..., n-1\}$ and $\oplus$ the operation of addition of the elements in $\mathbb{Z}_n$ modulo n. Then $< \mathbb{Z}_n, \oplus >$ is a commutative group.*

For simplification the group $< \mathbb{Z}_n, \oplus >$ as described above, from now on will be denoted by $< \mathbb{Z}_n, + >$. Another example of a group is $< \mathbb{Z}, + >$ which is the set of all integers $\mathbb{Z}$ with the operation of normal addition, whereas $< \mathbb{Z}, \cdot >$ is not a group since the element 0 does not have an inverse under normal multiplication. Likewise, the sets of rational numbers $\mathbb{Q}$ and real numbers $\mathbb{R}$ form commutative groups under normal addition but do not form groups under multiplication. Contrary to that, $< \mathbb{Q}^*, \cdot >$ and $< \mathbb{R}^*, \cdot >$ are groups. Commutative groups with an addition-like operation are also called **Abelian** groups, named after the mathematician Niels Henrik Abel.

Even though groups are useful in various of areas, they are limited due to the fact that they only have one operation associated with them. This brings us into introducing another algebraic structure, called a **ring**.

Definition 1.4. *A **ring** $< R, \oplus, * >$ is a set R of objects with two binary operations $\oplus$ (addition) and $*$ (multiplication) which satisfy the following properties.*

1. *R is closed under addition and multiplication, i.e., $\forall a, b \in R$, $(a + b) \in R$ and $(a * b) \in R$*
2. *$< R, \oplus >$ forms a commutative group. The additive identity is usually denoted by 0.*
3. *Multiplication is associative, i.e., $\forall a, b, c \in R$ it holds that $(a * b) * c = a * (b * c)$.*
4. *The left and right distributive laws hold, i.e., $\forall a, b, c \in R$, it holds that:*

$$a * (b \oplus c) = a * b \oplus a * c, \; (a \oplus b) * c = a * c \oplus b * c.$$

A ring $< R, \oplus, * >$ is said to be **commutative** if $a*b = b*a$, $\forall a, b \in R$. A ring $< R, \oplus, * >$ is said to be a **ring with identity**, if there exists an element e such that $a * e = e * a = a$, $\forall a \in R$. The identity element in this case is usually denoted by 1. As an example, let $< \mathbb{Z}_2, \oplus, * >$ denote the addition and multiplication of the numbers $\{0, 1\}$ modulo 2. Then $< \mathbb{Z}_2, \oplus, * >$ forms a ring with the following addition and multiplication tables.

$\oplus$	0	1
0	0	1
1	1	2

$*$	0	1
0	0	0
1	0	1

Similarly, the addition and multiplication tables for the rings $< \mathbb{Z}_3, \oplus, * >$ and $< \mathbb{Z}_4, \oplus, * >$ are shown below.

$\oplus$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$*$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

$\oplus$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$*$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

Note that none of the ring definitions above require that there exists a multiplicative inverse of the elements in the ring. As an example, in $< \mathbb{Z}_4, \oplus, * >$, the element 2 does not have a multiplicative inverse, whereas the element 3 does have an inverse since $3 * 3 = 1$.

Likewise as we did in groups, we will formalize that the set $\mathbb{Z}_n$ forms a ring under addition and multiplication modulo n in the theorem that follows.

Theorem 1.5. *Let $\mathbb{Z}_n$, with $n \geq 2$, be the set $\{0, 1, 2, ..., n-1\}$ and $\oplus$ and $*$ be the operations of addition and multiplication, respectively, of the elements in $\mathbb{Z}_n$ modulo n. Then, $< \mathbb{Z}_n, \oplus, * >$ is a commutative ring with identity.*

For simplification, from now on, the ring $< \mathbb{Z}_n, \oplus, * >$ as described above, will be denoted by $< \mathbb{Z}_n, +, \cdot >$.

Other examples of rings include $< \mathbb{Z}, +, \cdot >$, $< \mathbb{Q}, +, \cdot >$, and $< \mathbb{R}, +, \cdot >$, that is, the sets of integers, rational numbers, and real numbers, respectively, under the usual rules of addition and multiplication. All of the examples mentioned above refer to commutative rings. If $\mathbb{M}$ is the set of all 2-by-2 matrices, then $< \mathbb{M}, +, \cdot >$ is a ring with identity $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. In this case, multiplicative commutativity does not hold.

Definition 1.5. *A **field** $< \mathbb{F}, +, \cdot >$ is a set $\mathbb{F}$ of objects with two binary operations $+$ (addition) and $\cdot$ (multiplication), which satisfy the following properties.*

1. *$\mathbb{F}$ is closed under addition and multiplication, i.e., $\forall a, b \in \mathbb{F}$, $(a + b) \in \mathbb{F}$ and $(a \cdot b) \in \mathbb{F}$.*

2. *Additive identity: There exists an element in* $\mathbb{F}$*, which we denote by* 0*, such that* $a + 0 = 0 + a = a$*,* $\forall a \in \mathbb{F}$*.*

3. *Multiplicative identity: There exists a non zero element in* $\mathbb{F}$*, which we denote by* 1*, such that* $a \cdot 1 = 1 \cdot a = a$*,* $\forall a \in \mathbb{F}$*.*

4. *Additive Inverse: For every* $a \in \mathbb{F}$*, there exist an element* b *in* $\mathbb{F}$*, such that* $a + b = 0$*,* b *is usually denoted as* $-a$*.*

5. *Multiplicative Inverse: For every* $a \in \mathbb{F}$*, with* $a \neq 0$ *there exist an element* b *in* $\mathbb{F}$*, such that* $a \cdot b = 1$*,* b *is usually denoted as* a^{-1}*.*

6. *Addition and multiplication are associative, i.e.,* $(a + b) + c = a + (b + c)$ *and* $(a \cdot b) \cdot c = a \cdot (b \cdot c)$*,* $\forall a, b \in \mathbb{F}$*.*

7. *Addition and multiplication are commutative, i.e.,* $a + b = b + a$*, and* $a \cdot b = b \cdot a$*,* $\forall a, b \in \mathbb{F}$*.*

8. *Multiplication distributes over addition, i.e.,* $a \cdot (b + c) = a \cdot b + a \cdot c$*,* $\forall a, b, c \in \mathbb{F}$*.*

That is, the set $\mathbb{F}$ with the addition operation $+$ forms an Abelian group, the additive identity is 0. The set $\mathbb{F}\backslash\{0\}$ with the multiplication operation $\cdot$ forms a commutative group, where the identity is 1. Furthermore, multiplication distributes over addition. Fields have a similar structure with rings, they differ in that rings don't require the existence of a multiplicative inverse; in fact, they might not have a multiplicative identity. Every field is a ring, but rings are not always fields.

The set of real numbers $\mathbb{R}$ under the operations of normal addition and multiplication is the most commonly known field. Another example of a field is the ring $< \mathbb{Q}, +, \cdot >$. Whereas, the ring $< \mathbb{Z}, +, \cdot >$ does not form a field, since there does not exist a multiplicative inverse for every element in $< \mathbb{Z}, +, \cdot >$.

1.3 Galois Fields

Definition 1.6. *If, for a field* $< \mathbb{F}, +, \cdot >$*,* $|\mathbb{F}|$ *is an integer* q*, then the field is said to be a* ***finite field*** *or a* ***Galois field****, named after Évariste Galois, and is denoted by GF*(q)*.*

It is possible to construct such a field if q is a prime number or a power of a prime. Below we start off by showing how to construct a field GF(p) for a prime p and then how to extend the field GF(p) into a "bigger" field GF(p^m).

If p is prime, then $\mathbb{F} = \{0, 1, ..., p-1\}$ and the addition and multiplication operations are defined as normal addition and multiplication modulo p. That is, the ring $< \mathbb{Z}_p, +, \cdot >$ for p prime forms a finite field. For example, the ring $< \mathbb{Z}_2, +, \cdot >$ forms the field GF(2) with the following addition and multiplication tables.

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

The ring $< \mathbb{Z}_3, +, \cdot >$ forms GF(3). The corresponding addition and multiplication tables are as follows.

$+$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\cdot$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Note that, since 4 is not a prime number, the ring $< \mathbb{Z}_4, +, \cdot >$, with normal addition and multiplication modulo 4, does not form a field. As it can be seen in the multiplication table below, the element 2 does not have a multiplicative inverse, therefore the ring $< \mathbb{Z}_4, +, \cdot >$ does not form a field.

$\oplus$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$*$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

The ring $< \mathbb{Z}_5, +, \cdot >$ with the following addition and multiplication tables forms GF(5).

$+$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$\cdot$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

All of the above can be formalized in the following theorem.

Theorem 1.6. *If p is a prime number then the ring $< \mathbb{Z}_p, +, \cdot >$ is a (Galois) field.*

Proof. We have already established in *Theorem 1.5* that $< \mathbb{Z}_p, +, \cdot >$ forms a commutative ring with identity. The key remaining requirement to show that it is also a field is to prove that every element in the set $\mathbb{Z}_p \backslash \{0\}$ has a multiplicative inverse.

Let a be a non zero element in $\mathbb{Z}_p$. Form the set $A = \{1 \cdot a, 2 \cdot a, ..., (p-1) \cdot a\}$, that is, the set of all the nonzero elements of $\mathbb{Z}_p$ multiplied by a. We will show that every element in the set A is distinct, which would imply that A contains all non zero elements of $\mathbb{Z}_p$. Thus, the element $1 \in A$, which implies that the element a does indeed have a multiplicative inverse.

Let $m \cdot a$ and $k \cdot a$ be two elements of the set A, with $m \cdot a = k \cdot a$. Then, $m \cdot a = k \cdot a \implies m \cdot a - k \cdot a = 0 \implies (m-k) \cdot a = 0$. If $m \neq k$, for the last equation to hold, it must be the case that p divides $(m-k) \cdot a$. But that can't be true since p is prime and both of the elements $(m-k), a \in \mathbb{Z}_p \backslash \{0\}$ are relatively prime to p and therefore do not share any common divisors with p. Hence, for two elements $m \cdot a$ and $k \cdot a$ in A, with $m \neq k$, it holds that $m \cdot a \neq k \cdot a$, therefore all elements in A are distinct. □

It is possible to extend the field GF(p) to a field GF(p^m). In this case, GF(p) is called the **base field** and GF(p^m) the **extension field**. Any element $b \in \text{GF}(p^m)$ can be represented as an m-tuple $b = (b_1, b_2, ..., b_m)$ with $b_i \in \text{GF}(p)$ for $i = 1, 2, ..., m$, or it can be represented in a polynomial form as $b(\alpha) = b_1 + b_2\alpha + ... + b_m\alpha^{m-1}$. Below follows a table of all the elements of $GF(2^3)$ in their different representations.

Polynomial Representation	Vector Representation	Integer Representation
0	000	0
1	100	1
α	010	2
α^2	001	4
$1 + \alpha$	110	3
$\alpha + \alpha^2$	011	6
$1 + \alpha^2$	101	5
$1 + \alpha + \alpha^2$	111	7

The addition in the extension field is defined as follows. Let $b = (b_1, b_2, ..., b_m)$, $c = (c_1, c_2, ..., c_m) \in \text{GF}(q^m)$, then $b + c = (b_1 + c_1, b_2 + c_2, ..., b_m + c_m)$. Equivalently, in polynomial form, $b(\alpha) + c(\alpha) = (b_1 + c_1) + (b_2 + c_2)\alpha + ... + (b_m + c_m)\alpha^{m-1}$, where in both cases the addition of b_i and c_i is done as defined in the base field. The addition tables of GF(8) elements in vector and integer, respectively, representations can be seen below.

+	000	100	010	001	110	011	101	111
000	000	100	010	001	110	011	101	111
100	100	000	110	101	010	111	001	011
010	010	110	000	011	100	001	111	101
001	001	101	011	000	111	010	100	110
110	110	010	100	111	000	101	011	001
011	011	111	001	010	101	000	110	100
101	101	001	111	100	011	110	000	010
111	111	011	101	110	001	100	010	000

+	0	1	2	4	3	6	5	7
0	0	1	2	4	3	6	5	7
1	1	0	3	5	2	7	4	6
2	2	3	0	6	1	4	7	5
4	4	5	6	0	7	2	1	3
3	3	2	1	7	0	5	6	4
6	6	7	4	2	5	0	3	1
5	5	4	7	1	6	3	0	2
7	7	6	5	3	4	1	2	0

Before defining multiplication, the following definition is needed.

Definition 1.7. *A nonconstant polynomial $p(x)$ with coefficients over GF(q) (i.e., $p(x) \in GF(q)[x]$) is said to be irreducible over GF(q) if and only if it cannot be factored to a product $f(x)g(x)$ where both $f(x), g(x)$ are polynomials in $GF(q)[x]$ and are of degree less than the degree of $p(x)$.*

To define multiplication in $\mathrm{GF}(p^m)$, it is necessary to find an irreducible polynomial $p(\alpha) = p_0 + p_1\alpha + ... + p_m\alpha^m$ over $\mathrm{GF}(p)$. Then, the multiplication between two elements $b, c \in \mathrm{GF}(q^m)$ is defined as

$$b(\alpha)c(\alpha) \mod p(\alpha)$$

where each individual addition and multiplication is done as defined in the base field $\mathrm{GF}(q)$. As an example, let $p(\alpha) = 1 + \alpha + \alpha^3$ be an irreducible polynomial over $\mathrm{GF}(2)$. Let $b = (1, 0, 1) = 1 + \alpha^2$ and $c = (1, 1, 0) = 1 + \alpha$. Then $b \cdot c = b(\alpha)c(\alpha) \mod p(\alpha) = (1 + \alpha^2)(1 + \alpha) \mod p(\alpha) = 1 + \alpha + \alpha^2 + \alpha^3 \mod p(\alpha) = \alpha^2 = (0, 0, 1)$.

Now we will show that the extension $\mathrm{GF}(p^m)$ of a field $\mathrm{GF}(q)$, as described above, is indeed a field. Most of the properties of the definition of the field are straightforward to show. Let b,c,d be some elements in $\mathrm{GF}(p^m)$ with
$b = (b_1, b_2, ..., b_m)$, $b(a) = b_1 + b_2a + ... + b_ma^{m-1}$,
$c = (c_1, c_2, ..., c_m)$, $c(a) = c_1 + c_2a + ... + c_ma^{m-1}$,
$d = (d_1, d_2, ..., d_m)$, $d(a) = d_1 + d_2a + ... + d_ma^{m-1}$.
Also, let $p(a)$ be the m-th degree irreducible polynomial used to define multiplication. Then, it holds that:

- $b + c = (b_1 + c_1, b_2 + c_2, ..., b_m + c_m) = (k_1, k_2, ...k_m) = k$ with $k \in \mathrm{GF}(p^m)$ as an m-tuple with $k_i = (b_i + c_i) \in \mathrm{GF}(p)$. Therefore, $\mathrm{GF}(p^m)$ is closed under addition.

- $b(a) \cdot c(a) = b(a)c(a) \mod p(a) = k(a)$ where $k(a)$ is a polynomial with coefficients in $\mathrm{GF}(p)$ and of degree less than m. Therefore, $k \in \mathrm{GF}(p^m)$ and $\mathrm{GF}(p^m)$ is closed under multiplication.

- The additive identity is the m-tuple $(0, 0, ..., 0)$, which we will denote as simply 0: $0 + b = b + 0 = (b_1 + 0, b_2 + 0, ..., b_m + 0) = (b_1, b_2, ..., b_m) = b$.

- The multiplicative identity, denoted by 1, is the constant polynomial 1: $1 \cdot b = b \cdot 1 = b(a)1 \mod p(a) = b(a) \mod p(a) = b$.

- The additive inverse of an element b, denoted by $-b$, is the element $-b = (-b_1, -b_2, ..., -b_m)$ (where $-b_i$ is the additive inverse of the element b_i in $\mathrm{GF}(p)$): $b - b = (b_1 - b_1, b_2 - b_2, ..., b_m - b_m) = (0, 0, ..., 0) = 0$.

- To show that any non zero element b in $\mathrm{GF}(p^m)$ does have a multiplicative inverse, we will once again form the set $A = \{k \cdot b : k \in \mathrm{GF}(p^m), k \neq 0\}$ and show that the elements in A are distinct and thus the multiplication identity element 1 must be in A. Let $t \cdot b$ and $s \cdot b$ be two elements in A with $t \cdot b = s \cdot b$. Then, it holds that: $t \cdot b = s \cdot b \implies t \cdot b - s \cdot b = 0 \implies (t - s) \cdot b = 0$.
  Substitute $(t - s)$ with d and note that d is in $\mathrm{GF}(p^m)$. Rewrite the above relation as:

$(t-s)\cdot b = 0 \implies d\cdot b = 0 \implies d(a)b(a) \mod p(a) = 0$.
Since $b(a) \neq 0$, if also $d(a) \neq 0$, then, for the above equation to hold, it must be the case that $p(a)$ divides $d(a)b(a)$. But $p(a)$ is an irreducible polynomial of m-th degree and $b(a)$ is a polynomial of degree less than m, which means that $\gcd(p(a), b(a)) = 1$ and, hence, it must be the case that $p(a)$ divides $d(a)$. But $p(a)$ cannot divide $d(a)$ because $\gcd(p(a), d(a)) = 1$ (for the same reasoning as above). Thus, it must be the case $d(a) = 0$ which yields $d(a) = 0 \implies d = 0 \implies t - s = 0 \implies t = s$. Thus, we have proved that the elements in the set A are distinct and, therefore, every nonzero element a in $\mathrm{GF}(p^m)$ does have a multiplicative inverse.

- Addition is commutative:
  $b + c = (b_1 + c_1, b_2 + c_2, ..., b_m + c_m) = (c_1 + b_1, c_2 + b_2, ..., c_m + b_m) = c + b$.

- Addition is associative:
  $(b + c) + d = ((b_1 + c_1) + d_1, (b_2 + c_2) + d_2, ..., (b_m + c_m) + d_m) = (b_1 + (c_1 + d_1), b_2 + (c_2 + d_2), ..., b_m + (c_m + d_m)) = b + (c + d)$.

- Multiplication is commutative:
  $b \cdot c = b(a)c(a) \mod p(a) = c(a)b(a) \mod p(a) = c \cdot b$.

- Multiplication is associative:
  $(b\cdot c)\cdot d = (b(a)c(a) \mod p(a))d(a) \mod p(a) = b(a)c(a)d(a) \mod p(a) = [b(a)[c(a)d(a) \mod p(a)]] \mod p(a) = b \cdot (c \cdot d)$.

- Multiplication distributes over addition: $b \cdot (c + d) = b \cdot c + b \cdot d$.
  Here we take advantage of the division algorithm for polynomials. That is, we can write a polynomial $a(x)$ as:
  $a(x) = q(x)d(x) + r(x)$ with $a(x) \mod d(x) = r(x)$ and $\deg(r(x)) < \deg(d(x))$.
  It holds that $b \cdot (c + d) = b(a)(c(a) + d(a)) \mod p(a)$. Applying the division algorithm on $b(a)(c(a) + d(a))$ yields:
  $b(a)(c(a) + d(a)) = q_1(a)p(a) + r_1(a) \implies$
  $r_1(a) = b(a)(c(a) + d(a)) - q_1(a)p(a)$, with $\deg(r_1(x)) < \deg(p(x)) = m$. Thus, we can write:
  $b(a)(c(a)+d(a)) \mod p(a) = r_1(a) = b(a)(c(a)+d(a)) - q_1(a)p(a) = \sum_{i=1}^{m} b_i a^{i-1} (\sum_{j=1}^{m} c_j a^{j-1} + d_j a^{j-1}) - q_1(a)p(a) =$
  $\sum_{i=1}^{m}\sum_{j=1}^{m} b_i c_j a^{i+j-2} + \sum_{i=1}^{m}\sum_{j=1}^{m} b_i d_j a^{i+j-2} - q_1(a)p(a) =$
  $b(a)c(a) + b(a)d(a) - q_1(a)p(a)$.
  In the step before the last one, we took advantage of the fact that the operations are in the base field and thus multiplication distributes over addition. Thus, $r_1(a) = b(a)c(a) + b(a)d(a) - q_1(a)p(a)$.

  We apply the division algorithm again on $b(a)c(a)$ and on $b(a)d(a)$, which yields:
  $b(a)c(a) = q_2(a)p(a) + r_2(a)$, with $\deg(r_2(x)) < \deg(p(x))$
  and $b(a)d(a) = q_3(a)p(a) + r_3(a)$, with $\deg(r_3(x)) < \deg(p(x))$.

Now, $b \cdot c + b \cdot d$ can be written as:
$b \cdot c + b \cdot d = (b(a)c(a) \mod p(a)) + (b(a)d(a) \mod p(a)) = r_2(a) + r_3(a) = b(a)c(a) - q_2(a)p(a) + b(a)d(a) - q_3(a)p(a) \implies$
$r_2(a) + r_3(a) = b(a)c(a) + b(a)d(a) - q_2(a)p(a) - q_3(a)p(a)$. In the last equation we can substitute $(b(a)c(a) + b(a)d(a))$ with $r_1 + q_1(a)p(a)$:
$r_2(a) + r_3(a) = r_1 + q_1(a)p(a) - q_2(a)p(a) - q_3(a)p(a) \implies$
$r_2(a) + r_3(a) - r_1(a) = (q_1(a) - q_2(a) - q_3(a))p(a) \implies r(a) = q(a)p(a)$.

Now note that the above relation can only hold if the degree of the polynomial on the left hand side is the same as the degree of the polynomial on the right side. On the left hand side, if $r(a) \neq 0$ we have a polynomial of degree less than m. While on the right side, since $p(a)$ is of degree m and is multiplied by the polynomial $q(a)$, if $q(a) \neq 0$ the degree becomes greater or equal to m. And thus the above relation cannot be true. Therefore, it must be the case that $q(a) = 0$ and $r(a) = 0 \implies r_1(a) = r_2(a) + r_3(a) \implies b(a)(c(a) + d(a)) \mod p(a) = (b(a)c(a) \mod p(a)) + (b(a)d(a) \mod p(a)) \implies b \cdot (c + d) = b \cdot c + b \cdot d$. Thus, we have proven that multiplication distributes over addition.

We have shown that GF(p^m) satisfies all the properties of the field definition, therefore GF(p^m) is a field. An extended field GF(p^m) can be further extended into a field GF((p^m)^n)).

Definition 1.8. *Let a be a non zero element in GF(q), the **order** of a, denoted by ord(a), is the smallest positive integer n such that $a^n = \underbrace{a \cdot a \cdot ... \cdot a}_{n} = 1$.*

For example, we calculate the order of 4 and 2 in $GF(5)$. For $4 \in$ GF(5), $4^1 = 4$, $4^2 = 1$, therefore ord(4) = 2. For $2 \in$ GF(5), $2^1 = 2$, $2^2 = 4$, $2^3 = 3$, $2^4 = 1$, therefore ord(2) = 4.

Definition 1.9. *An element $a \in GF(q)$ is said to be a **primitive** element if and only if ord$(a) = q - 1$.*

The powers $a^1, a^2, ...a^{q-1}$ of a primitive element $a \in$ GF(q) generate all the non zero elements of GF(q). In the previous example, it can be seen that the element 2 in GF(5) is of order 4, therefore it is a prime element. In the same example, it is also clear that the powers $2^1, 2^2, 2^3, 2^4$ generate all the non zero elements of GF(5).

Theorem 1.7. *[2] For a Galois field GF(q), if $t|q - 1$, then there exist $\phi(t)$ elements of order t in GF(q), where $\phi(t)$ is the Euler Totient function.*

As a direct corollary we get the following theorem.

Theorem 1.8. *There exist $\phi(q - 1)$ primitive elements in GF(q).*

The existence of a primitive element enables a new representation of the nonzero Galois field elements, called the **power representation**. As an example, we present the GF(2^3) elements. The element $b = \alpha$ is a primitive element.

Polynomial Representation	Vector Representation	Integer Representation	Power Representation ($b = \alpha$)
0	000	0	-
1	100	1	b^0
α	010	2	b^1
α^2	001	4	b^2
$1+\alpha$	110	3	b^3
$\alpha+\alpha^2$	011	6	b^4
$1+\alpha^2$	101	5	b^5
$1+\alpha+\alpha^2$	111	7	b^6

Then, the multiplication table of GF(8) elements in power representation is as follows.

$\cdot$	0	1	α^1	α^2	α^3	α^4	α^5	α^6
0	0	0	0	0	0	0	0	0
1	0	1	α^1	α^2	α^3	α^4	α^5	α^6
α^1	0	α^1	α^2	α^3	α^4	α^5	α^6	1
α^2	0	α^2	α^3	α^4	α^5	α^6	1	α^1
α^3	0	α^3	α^4	α^5	α^6	1	α^1	α^2
α^4	0	α^4	α^5	α^6	1	α^1	α^2	α^3
α^5	0	α^5	α^6	1	α^1	α^2	α^3	α^4
α^6	0	α^6	1	α^1	α^2	α^3	α^4	α^5

Definition 1.10. *Let $b \in GF(q^m)$.*
*a) The **conjugates** of b with respect to the subfield GF(q) are the elements: $b, b^q, b^{q^2}, b^{q^3}, ...$*
*b) The conjugates of b with respect to GF(q) form a set called the **conjugacy class** of b with respect to GF(q).*

For example, let a be a primitive element in GF(2^3). The conjugates of a with respect to GF(2) are

$$a, a^2, a^4, a^8 = a.$$

Now, let $b = a^3$. The conjugates of b with respect to GF(2) are

$$b = a^3, b^2 = a^6, b^4 = a^{12} = a^5, b^8 = a^{24} = a^3.$$

The only other elements left in GF(2^3) are 0 and 1, each of them always form their own conjugacy class. Hence, the conjugacy classes of the elements of GF(2^3) with respect to GF(2) are

$$\{0\}, \{1\}, \{a, a^2, a^4\}, \{a^3, a^5, a^6\}.$$

Definition 1.11. *Let $b \in GF(q^m)$. The **minimal polynomial** of b with respect to GF(q) is the smallest-degree, nonzero, monic polynomial $M(x) \in GF(q)[x]$ such that $M(b) = 0$.*

Theorem 1.9. *[2] Let $A \subseteq GF(q^m)$ be a conjugacy class with respect to GF(q). Then $M(x) = \prod_{b \in A} (x - b)$ is a minimal polynomial of b with respect to GF(q) for every $b \in A$.*

For example, the following table shows the conjugacy classes of GF(8) with respect to GF(2) and corresponding minimal polynomials with respect to GF(2).

Conjugacy Class	Minimal Polynomial
$\{0\}$	$M(x) = x$
$\{1\}$	$M_0(x) = x + 1$
$\{\alpha, \alpha^2, \alpha^4\}$	$M_1(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^4) = x^3 + x + 1$
$\{\alpha^3, \alpha^6, \alpha^5\}$	$M_3(x) = (x - \alpha^3)(x - \alpha^6)(x - \alpha^5) = x^3 + x^2 + 1$

Chapter 2

Designing BCH codes

Transmitting information over a medium includes the risk of the information to be altered. In order to retrieve the original information-message we make use of **error correction algorithms**, which aim into developing ways of correcting the altered message (Illustration in Fig. 2.1). Such mediums, referred to as **channels**, may be electrical wires, the air in the case of wireless communications, or even storage mediums like a hard drive. A message that has been altered is said to be a message containing **errors**. In this section, we deal with **error correction coding** more specifically we show how to design a BCH code (Bose–Chaudhuri–Hocquenghem code).

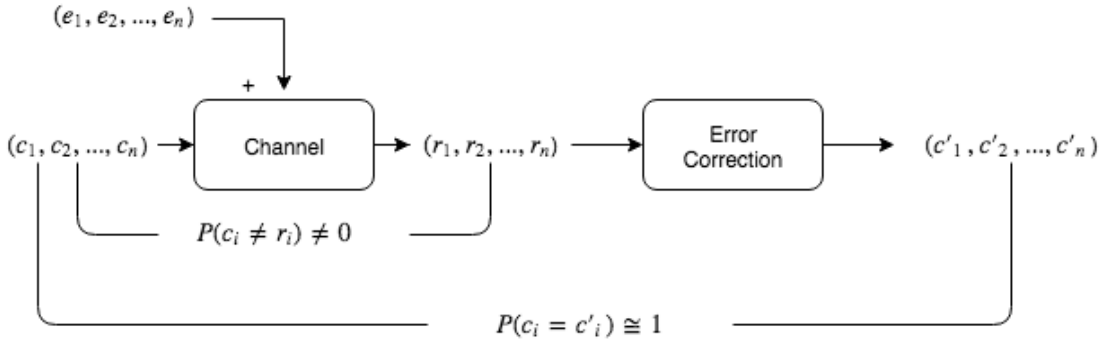

Figure 2.1: Channel adds errors to the transmitted codeword.

Definition 2.1. *An (n, k) **block code** C over an alphabet of q symbols is a set of q^k n-vectors called **codewords**. Associated with the code is an encoder which maps a message k-tuple* $\mathbf{m}$ *to its associated codeword.*

Definition 2.2. *A block code C over a field $\mathbb{F}_q$ of q symbols of length n and q^k codewords is a q-ary **linear** (n, k) code if and only if its q^k codeword form a k-dimensional vector subspace of the vector space of all the n-tuples $\mathbb{F}_q^n$. The number n is said to be the **length** of the code and the number k is the **dimension** of the code.*

Definition 2.3. *A (n, k) block code C is said to be **cyclic**, if it is linear and, for every codeword $c = (c_0, c_1, ..., c_{n-1})$ in C, its right cyclic shift $c' = (c_{n-1}, c_0, c_1, ..., c_{n-2})$ is also in C.*

It can be shown that an (n, k) cyclic code has a unique minimal monic polynomial $g(x)$, called the **generator polynomial**, such that every code polynomial in the code can be expressed as a multiple of the generator and the message:

$$c(x) = m(x)g(x).$$

BCH codes are cyclic codes and thus can be specified by a generator polynomial $g(x)$. The process of calculating $g(x)$ for a BCH code over GF(q) is described below.

Let q be the number of different symbols over the code alphabet, under the restriction that q is a power of a prime number. Let n be the code length, under the restriction that n and q are relatively prime, and t be the error correction capability. Begin by finding the smallest m such that $n|q^m - 1$. Since $n|q^m - 1$ from *Theorem 1.7*, there exist $\phi(n)$ elements of order n in GF(q^m). Let one of them be β, then $\beta^n = 1$.

For example, let $n = 85$, $q = 2$. By some trial and error we find that the smallest m such that $n|q^m - 1$ is $m = 8$, since $85|2^8 - 1 \implies 85|255$. Now, let α be a primitive element in GF(2^8) = GF(256). The element $\beta = \alpha^3$ is of order 85. It holds that $\beta^{85} = (\alpha^3)^{85} = \alpha^{255} = 1$.

Another example is the following. Let $n = 15$, $q = 2$. Then, the smallest m such that $n|q^m - 1$ is $m = 4$, since $q^m - 1 = 2^4 - 1 = 15$. Let α be a primitive element in GF(2^4) = GF(16). Then, $\beta = \alpha$ is of order 15. It holds that $\beta^{15} = \alpha^{15} = 1$.

Now, take $2t$ consecutive powers of β, i.e., $\beta, \beta^2, ..., \beta^{2t}$, and determine the minimal polynomials with respect to GF(q) of each of these powers of β.

In the previous example with $q = 2$, $n = 15$, and $m = 4$, $\beta = \alpha$ is an element of order 15 in GF(16). Furthermore, let $t = 2$ and $b = 1$. Then, the $2t$ consecutive powers of β are $\beta^1, \beta^2, \beta^3, \beta^4$. By finding the conjugacy classes with respect to GF(2) of these powers, we obtain the conjugacy class of $\beta^1, \beta^2, \beta^4$ which is $A = \{\beta^1, \beta^2, \beta^4, \beta^8\}$ and the conjugacy class of β^3 which is $B = \{\beta^3, \beta^6, \beta^{12}, \beta^9\}$. Thus, the minimal polynomials of $\beta^1, \beta^2, \beta^3, \beta^4$ with respect to GF(2) are:
$m_A(x) = (x - \beta^1)(x - \beta^2)(x - \beta^4)(x - \beta^8) = x^4 + x + 1$,
$m_B(x) = (x - \beta^3)(x - \beta^6)(x - \beta^{12})(x - \beta^9) = x^4 + x^3 + x^2 + x + 1$.
The generator polynomial $g(x)$ is calculated as the multiple of these minimal polynomials, that is, $g(x) = m_A(x)m_B(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1) = x^8 + x^7 + x^6 + x^4 + 1$. The code is a $(n, n - \deg(g(x)))$ cyclic code.

The minimal polynomials were taken with respect to GF(q), therefore $g(x) \in$ GF(q)[x]. Another effect of using the minimal polynomials with respect to GF(q) is that $g(x)$ might have more than $2t$ roots, hence the code might exceed the error capability of t. For encoding purposes, it is sufficient to work on the "small field" GF(q). Decoding, however, requires operations in the "big field" GF(q^m).

Chapter 3

Reed-Solomon codes

3.1 Designing a Reed-Solomon code

Reed-Solomon (RS) codes are a family of cyclic error correction codes, they were introduced by Irving S. Reed and Gustave Solomon in 1960 [3]. Despite of them being some of the oldest error correction codes, they are widely used today in communications, in data storage systems and other applications such as QR codes and barcodes. As we will see in this chapter, Reed-Solomon codes are closely related with BCH codes. We will show how to construct an RS code and how encoding and decoding is done.

Definition 3.1. *A **Reed-Solomon** code is defined as a q-ary BCH code of length $n = q-1$.*

The design of an RS code is straightforward, from definition we have $n = q - 1$. The smallest m such that $n|q^m - 1$ is $m = 1$. So the "big field" GF(q^m) is GF(q), in which we are looking for an element of order $n = q-1$, such an element can be any primitive element in GF(q). In GF(q), the minimal polynomial with respect to GF(q) of any element $\beta \in$ GF(q) is simply $(x - \beta)$. Therefore the generator polynomial for an RS code is:

$$g(x) = (x - \alpha)(x - \alpha^2)...(x - \alpha^{2t})$$

where α is a primitive element in GF(q).

As an example, consider a double error correction ($t = 2$) RS code of length $n = 2^4 - 1 = 15$ over GF(2^4). Let α be a primitive element in GF(2^4), the code generator polynomial $g(x)$ has $\alpha^1, \alpha^2, \alpha^3, \alpha^4$ as roots. The generator polynomial is:

$$g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4) \implies g(x) = x^4 + \alpha^{13}x^3 + \alpha^6x^2 + \alpha^3x + \alpha^{10}$$

In Reed-Solomon codes the degree of $g(x)$ is always equal to $2t$. thus $n - k = 2t$. The design distance is $\delta = n - k + 1$ which means Reed-Solomon codes achieve the singleton bound and hence are maximum distance separable(MDS) codes.

In the following table we compare the design of BCH and RS codes.

	BCH code over GF(q)	RS code over GF(q)
Length n	$n\|q^m - 1$	$n = q - 1$
Element of order n	β an element in GF(q^m) such that ord(β) = n	α a primitive element in GF(q) ord(α) = n
Minimal polynomials	$M_1(x), M_2(x), ...$	$(x - \alpha), (x - \alpha^2), ..., (x - \alpha^{2t})$
Generator Polynomial	$g(x) =$ $M_1(x)M_2(x)...$	$g(x) =$ $(x - \alpha)(x - \alpha^2)...(x - \alpha^{2t})$
$g(x)$ coefficients	$g(x)$ with coefficients in GF(q)	$g(x)$ with coefficients in GF(q)

3.2 Reed-Solomon Encoding

Reed-Solomon codes may be encoded just as any other cyclic code. The systematic encoding can be defined as:

$$c(x) = m(x)x^{n-k} - R_{g(x)}[m(x)x^{n-k}]$$

where $R_{g(x)}[p(x)]$ denotes the operation of taking the remainder after dividing $p(x)$ by $g(x)$. One of the advantages of the systematic encoding, is that, the codeword $c(x)$ contains the message $m(x)$.

To give an example of encoding, we use the generator polynomial $g(x)$ of the previous example, in integer representation

$$g(x) = x^4 + 13x^3 + 12x^2 + 8x + 7.$$

Let the message polynomial be

$$m(x) = 3 + x + 2x^2 + 3x^3 + 6x^4 + 14x^6 + 15x^7 + 7x^8 + 7x^9 + 5x^{10}.$$

This results into the codeword:

$$c(x) = 12 + 13x + 6x^2 + 14x^3 + \\ +3x^4 + x^5 + 2x^6 + 3x^7 + 6x^8 + 14x^{10} + 15x^{11} + 7x^{12} + 7x^{13} + 5x^{14}$$

In the example above it is clear that the message is included in the codeword.

3.3 Reed-Solomon Decoding

3.3.1 Syndromes

The algebraic decoding of Reed-Solomon codes has the following general steps:

- The syndrome computation.
- Finding the error locator polynomial.
- Calculation of the roots of the error locator polynomial.
- Calculation of the error values.

The syndromes indicate whether an error occurred or not. The next two steps are related with finding the locations, of the received word, where errors occurred. Having found the erroneous locations, in order to correct the errors, it is required to calculate the error values, which is the final step.

By design $g(a^1) = g(a^2) = ... = g(a^{2t}) = 0$, it follows that for a coderword $c(x)$:

$$c(a^1) = c(a^2) = ... = c(a^{2t}) = 0$$

Now, for a received polynomial $r(x) = c(x) + e(x)$, let:

$$S_j = r(a^j) = e(\alpha^j) = \sum_{k=0}^{n-1} e_k a^{jk},\ j = 1, 2, \ldots, 2t.$$

The values $S_1, S_2, ..., S_{2t}$ are called the syndromes of the received polynomial. The syndrome polynomial $S(x)$ is defined as

$$S(x) = S_1 + S_2 x + ... + S_{2t} x^{2t-1}.$$

If $S(x) = 0$, then the received polynomial is a codeword. As an example, consider that the received polynomial is the codeword $c(x)$ of the last example,
$c(x) = 12+13x+6x^2+14x^3+3x^4+x^5+2x^6+3x^7+6x^8+14x^{10}+15x^{11}+7x^{12}+7x^{13}+5x^{14}$
The syndromes can be calculated by evaluating $c(x)$ at $\alpha, \alpha^2, \alpha^3, \alpha^4$:
$S_1 = c(\alpha) = 0,$
$S_2 = c(\alpha^2) = 0,$
$S_3 = c(\alpha^3) = 0,$
$S_4 = c(\alpha^4) = 0.$
Now assume that, due to channel noise, two errors are added, on the coderword $c(x)$ and the received polynomial becomes
$r(x) = 12+13x+6x^2+14x^3+3x^4+x^5+2x^6+3x^7+6x^8+\underline{2x^9}+14x^{10}+\underline{x^{11}}+7x^{12}+7x^{13}+5x^{14}.$
Evaluating $r(x)$ at $\alpha, \alpha^2, \alpha^3, \alpha^4$ yields:
$S_1 = r(\alpha) = 12,$
$S_2 = r(\alpha^2) = 11,$
$S_3 = r(\alpha^3) = 4,$
$S_4 = r(\alpha^4) = 12.$
Thus we determine that the received polynomial contains errors.

3.3.2 Error Locator Polynomial

Suppose that $\mathbf{r}$ has v errors, at the locations $i_1, i_2, \ldots i_v$. Then:

$$S_j = \sum_{k=0}^{n-1} e_k a^{jk} = \sum_{l=1}^{v} e_{i_l} \alpha^{j i_l} = \sum_{l=1}^{v} e_{i_l} \alpha^{i_l j},\ j = 1, 2, \ldots, 2t$$

Now let $X_l = \alpha^{i_l}$, we can write $S_j = \sum_{l=1}^{v} e_{i_l} X_l^j,\ j = 1, 2, \ldots, 2t.$

If there was a way to calculate X_l, then we could also calculate i_l, which is by definition the location of the error. Hence, the X_l are called **error locators**. The **error locator polynomial** is defined as

$$\Lambda(x) = \prod_{l=1}^{v} (1 - X_l x) = \Lambda_0 + \Lambda_1 x + \cdots + \Lambda_v x^v$$

where $\Lambda_0 = 1$. The roots of $\Lambda(x)$ are the reciprocals of the error locators, since $\Lambda(X_k^{-1}) = 0$, for any $k = 1, 2, \ldots, v$. Thus, by finding the roots of the error locator polynomial, we can find the error locators. We aim at finding the coefficients of $\Lambda(x)$, the next relations is particularly helpful in doing so. The syndromes and the locator polynomial coefficients are related by the following equation:

$$\Lambda_v S_{j-v} + \Lambda_{v-1} S_{j-v-1} + \cdots + \Lambda_1 S_{j-1} + S_j = 0 \iff$$
$$S_j = -\sum_{i=1}^{v} \Lambda_i S_{j-i}, \text{ for } j = v+1, v+2, ..., 2t$$

Proof. Let X_l be an error locator, then:
$\Lambda(X_l^{-1}) = 0 \implies$
$\Lambda_v X_l^{-v} + ... + \Lambda_1 X_l^{-1} + \Lambda_0 = 0$
For $j = v+1, v+2, ..., 2t$, we can write:
$e_{i_l} X_l^j (\Lambda_v X_l^{-v} + ... + \Lambda_1 X_l^{-1} + \Lambda_0) = 0 \implies$
$e_{i_l} (\Lambda_v X_l^{j-v} + ... + \Lambda_1 X_l^{j-1} + \Lambda_0 X_l^j) = 0$
Summing the last relation over l yields:
$\sum_{l=1}^{v} e_{i_l} (\Lambda_v X_l^{j-v} + ... + \Lambda_1 X_l^{j-1} + \Lambda_0 X_l^j) = 0 \implies$
$\Lambda_v \sum_{l=1}^{v} e_{i_l} X_l^{j-v} + ... + \Lambda_1 \sum_{l=1}^{v} e_{i_l} X_l^{j-1} + \Lambda_0 \sum_{l=1}^{v} e_{i_l} X_l^j = 0 \implies$
$\Lambda_v S_{j-v} + ... + \Lambda_1 S_{j-1} + \Lambda_0 S_j = 0$
Since $\Lambda_0 = 0$ the last relation can be written as: $S_j = -\sum_{i=1}^{v} \Lambda_i S_{j-i}$ □

Below we present the Peterson-Gorenstein-Zierler algorithm. Which is one of many ways of calculating the error locator polynomial. The previous equation can be written in matrix form as follows.

$$\underbrace{\begin{bmatrix} S_1 & S_2 & ... & S_v \\ S_2 & S_3 & ... & S_{v+1} \\ S_3 & S_4 & ... & S_{v+2} \\ \vdots & & & \\ S_v & S_{v+1} & ... & S_{2v-1} \end{bmatrix}}_{M_v} \begin{bmatrix} \Lambda_v \\ \Lambda_{v-1} \\ \Lambda_{v-2} \\ \vdots \\ \Lambda_1 \end{bmatrix} = - \begin{bmatrix} S_{v+1} \\ S_{v+2} \\ S_{v+3} \\ \vdots \\ S_{2v} \end{bmatrix} \tag{3.1}$$

Since the number of actual errors v is not know in advance, the Peterson-Gorenstein-Zierler algorithm operates as follows. It starts off, by setting $v = t$ and forms the matrix M_v. If M_v is not invertible then v is reduced by 1, and then the new M_v matrix is formed. This is repeated until M_v is invertible. For an invertible matrix M_v, the equation 3.1 can be solved for the coefficients $\Lambda_1, \Lambda_2, ..., \Lambda_v$.

Algorithm 1: Peterson-Gorenstein-Zierler algorithm

Data: $S_1, S_2, ..., S_3$
Result: $\Lambda_1, \Lambda_2, ...\Lambda_v$
1 Set $v = t$ and form the matrix M_v
2 **while** M_v *not invertible* **do**
3 reduce v by one and form M_v again.
4 Solve for the coefficients $\Lambda_1, \Lambda_2, ..., \Lambda_v$

By continuing the last example, where the syndromes where calculated as $S_1 = 12, S_2 = 11, S_3 = 4, S_4 = 12$. For $v = t = 2$ the matrix $M_v = \begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} = \begin{bmatrix} 12 & 11 \\ 11 & 4 \end{bmatrix}$ is invertible.

By solving the equation

$$\begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} \begin{bmatrix} \Lambda_2 \\ \Lambda_1 \end{bmatrix} = - \begin{bmatrix} S_3 \\ S_4 \end{bmatrix} \implies \begin{bmatrix} 12 & 11 \\ 11 & 4 \end{bmatrix} \begin{bmatrix} \Lambda_2 \\ \Lambda_1 \end{bmatrix} = - \begin{bmatrix} 4 \\ 12 \end{bmatrix}.$$

We find $\Lambda_1 = 4, \Lambda_2 = 6$, so the output is the locator polynomial $\Lambda(x) = 1 + 4x + 6x^2$.

The Peterson-Gorenstein-Zierler algorithm involves straightforward linear algebra, as simple as it seems, there exist other algorithms that are computationally more efficient like the Sugiyama algorithm [4] and the Berlekamp-Massey algorithm [5].

3.3.3 Roots of the Error Locator Polynomial

The roots of the error locator polynomial are typically found with a process called *chien search*. Chien search is an exhaustive search over all of the nonzero elements of the finite field of interest GF(q). It operates by simply evaluating $\Lambda(x)$ at every nonzero element in the field, in succession, i.e., evaluating $\Lambda(x)$ at $x = 1, x = a, x = a^2, ..., x = a^{q-2}$. Which yields:

$$\begin{aligned} \Lambda(1) &= 1 + \Lambda_1 1 + \Lambda_2 1^2 + \cdots + \Lambda_v 1^v \\ \Lambda(a) &= 1 + \Lambda_1 a + \Lambda_2 a^2 + \cdots + \Lambda_v a^v \\ \Lambda(a^2) &= 1 + \Lambda_1 a^2 + \Lambda_2 (a^2)^2 + \cdots + \Lambda_v (a^2)^v \\ &\vdots \\ \Lambda(a^{q-2}) &= 1 + \Lambda_1 a^{q-2} + \Lambda_2 (a^{q-2})^2 + \cdots + \Lambda_v (a^{q-2})^v. \end{aligned}$$

The roots are the elements x_l for which $\Lambda(x_l) = 0$, $l = 1, 2, \ldots, v$. If a root $x_l = a^j$ is known, the error locator $X_l = x_l^{-1} = a^{i_l}$ can be calculated and thus the error can be located.

If the roots are not distinct, or lie in the wrong field, then the error pattern is said to be an *uncorrectable error pattern*, which results in a decoder failure. An uncorrectable error pattern may be observed if the error locator polynomial of degree v does not have v roots in GF(q).

Given the locator polynomial of the previous example, $\Lambda(x) = 1 + 4x + 6x^2 = 1 + a^2x + (a + a^2)x^2$. By utilizing the Chien search algorithm, we find the roots a^4, a^6.
$\Lambda(a^4) = 1 + a^2a^4 + (a + a^2)(a^4)^2 = 1 + a^6 + a^9 + a^{10} = 1000_2 + 0011_2 + 0101_2 + 1110_2 = 0$,
$\Lambda(a^6) = 0$.
The corresponding error locators are:
$X_1 = (a^4)^{-1} = a^{11} \implies i_1 = 11$,
$X_2 = (a^6)^{-1} = a^9 \implies i_2 = 9$.
Thus the errors are located on r_{11}, r_9 of the received polynomial. Now if we recall the $c(x)$ and $r(x)$ polynomials
$c(x) = 12+13x+6x^2+14x^3+3x^4+x^5+2x^6+3x^7+6x^8+14x^{10}+15x^{11}+7x^{12}+7x^{13}+5x^{14}$,
$r(x) = 12+13x+6x^2+14x^3+3x^4+x^5+2x^6+3x^7+6x^8+\underline{2x^9}+14x^{10}+\underline{x^{11}}+7x^{12}+7x^{13}+5x^{14}$,
we observe that, the error locations are indeed r_{11} and r_9.

3.3.4 Error Magnitudes

In order to correct the received polynomial, the error values must be determined. The error values are given by Forney's formula:

$$e_{i_l} = -\frac{\Omega(X_l^{-1})}{\Lambda'(X_l^{-1})}$$

where $\Omega(x) = S(x)\Lambda(x)$ ($\mod x^{2t}$) and $\Lambda'(x)$ is the formal derivative of $\Lambda(x)$.

As an example, let $S(x) = 12 + 11x + 4x^2 + 12x^3$, $\Lambda(x) = 1 + 4x + 6x^2$. Then:
$\Omega(x) = S(x)\Lambda(x) \pmod{x^{2t}} =$
$= (12 + 11x + 4x^2 + 14x^3)(1 + 4x + 6x^2)(\mod x^4) =$
$= (12 + 14x + 14x^4 + 14x^5) \pmod{x^4} =$
$= (12 + 14x)$
$\Lambda'(x) = (0 + 4 + (6 + 6)x) = 4.$
In the previous example we found that $X_1 = (a^4)^{-1}$ and $X_2 = (a^6)^{-1}$. Thus, the error values are:
$e_{i_1} = e_{11} = -\frac{\Omega(X_1^{-1})}{\Lambda'(X_1^{-1})} = -\frac{\Omega(a^4)}{\Lambda'(a^4)} = -\frac{12+14a^4}{4} = 14,$
$e_{i_2} = e_9 = -\frac{\Omega(X_2^{-1})}{\Lambda'(X_2^{-1})} = -\frac{\Omega(a^6)}{\Lambda'(a^6)} = 2.$
We now can form the error polynomial $e(x) = 2x^9 + 14x^{11}$. Finally we decide that,
$\hat{c}(x) = r(x) - e(x) =$
$12 + 13x + 6x^2 + 14x^3 + 3x^4 + x^5 + 2x^6 + 3x^7 + 6x^8 + \underline{2x^9} + 14x^{10} + \underline{x^{11}} + 7x^{12} + 7x^{13} +$
$5x^{14} - 2x^9 - 14x^{11} =$
$12+13x+6x^2+14x^3+3x^4+x^5+2x^6+3x^7+6x^8+14x^{10}+15x^{11}+7x^{12}+7x^{13}+5x^{14} = c(x),$
which is the error-free codeword.

Given the code length n, the message length k, the received polynomial $r(x)$, and the primitive element a used for the encoding. The decoding process can be summarized into Algorithm 2.

Algorithm 2: Reed-Solomon decoding algorithm

```
Data: n, k, α, r(x)
Result: m(x)
Calculate the syndrome polynomial S(x)
if S(x) = 0 then
    The coefficients of m(x) are the last k coefficients of r(x)
else
    Compute the error locator polynomial
    Apply the chien-search on the locator polynomial to find its roots.
    if succesful then
        Calculate the error magnitudes using Forney's formula
        Correct the received polynomial
        Retrieve the message polynomial m(x).
    else
        Declare decoding failure. Algorithm end.
Return m(x)
```

Chapter 4

Reed-Solomon Burst Error Decoding

Reed-Solomon code decoding competes favorably to other algorithms in burst error correction. Below we present a novel burst error correction algorithm by Yingquan Wu. The algorithm is able to correct a burst error of length f up to $d-2$, where $d = n-k+1$ is the design distance of the RS code. The decoding failure probability is bounded by $q^{-(d-3-f)}$ [6].

Definition 4.1. *Denote by* $\mathsf{B}(s, f, \mathbf{u})$ *a burst of errors* $\mathbf{u} = [u_0, u_1, ..., u_{f-1}]$ *that starts from* s *and has length* f*, such that:*

i. More than half the elements of the error vector $\mathbf{u}$ *are nonzero*

ii. The boundary locations s *and* $s+f-1$ *are erroneous i.e.,* $u_0 \neq 0$*,* $u_{f-1} \neq 0$*.*

iii. It is maximal in the sense that no super interval $[s', s+f'-1] \supset [s, s+f-1]$ *satisfies (i) and (ii)*

It is important to note that, if less than $\lceil \frac{f}{2} \rceil (< \lceil \frac{d-1}{2} \rceil)$ elements of $\mathbf{u}$ are nonzero, then the errors are more convenient to be treated as random errors. In which case, the conventional RS decoder is able to correct them. However, the conventional decoder fails to correct more than $\lceil \frac{d-1}{2} \rceil$ errors. On the other hand, given that the errors appear in bursts, the proposed algorithm can correct up to $d-2$ errors. Finally if $f \geq d-1$, then the proposed algorithm fails to correct the errors and the error pattern is said to be uncorrectable. Thus the limit on the burst error correction capability of an (n, k) RS code is $d-2$.

In order to correct the burst error, we will first try to locate it's position, which brings us in introducing the following theorem.

Theorem 4.1. *[6] Let* $\Lambda(x) = \prod_{i=0}^{d-3}(1 - a^{s'+i}x)$*. If a single burst* $\mathsf{B}(s, f, \mathbf{u})$ *has occurred such that the burst interval* $[s, s+f-1] \subseteq [s', s'+d-3]$ *then regardless of the actual burst error length* f*, the following holds.*

$$S_{d-2}\Lambda_0 + S_{d-3}\Lambda_1 + \cdots + S_0\Lambda_{d-2} = 0$$

A graphical illustration of *Theorem 4.1* is shown at Figure 2.1. Since *Theorem 4.1* holds true for any super-interval, with length $d-2$, of the actual burst interval we get the following theorem, which is illustrated in Fig. 4.2.

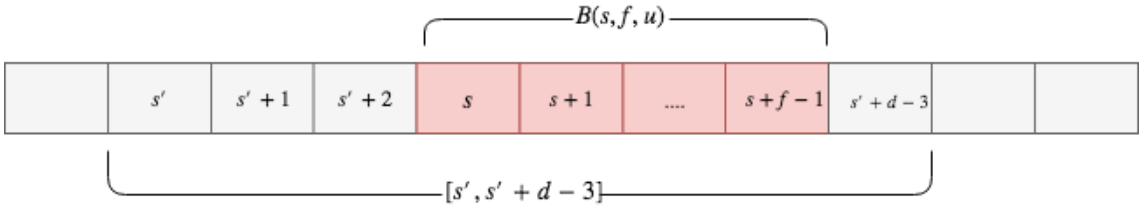

Figure 4.1: Graphical illustration of Theorem 4.1.

Theorem 4.2. *Let* $\Lambda^{(j)}(x) = \prod_{i=0}^{d-3}(1-a^{s+i-j}x)$ *for* $j = 0, 1, .., d-3$. *A single burst* $\mathsf{B}(s, f, \mathbf{u})$ *has occurred if and only if:*

$$\begin{aligned} S_{d-2}\Lambda_0^{(0)} + S_{d-3}\Lambda_1^{(0)} + \ldots S_0\Lambda_{d-2}^{(0)} &= 0 \\ S_{d-2}\Lambda_0^{(1)} + S_{d-3}\Lambda_1^{(1)} + \ldots S_0\Lambda_{d-2}^{(1)} &= 0 \\ &\vdots \\ S_{d-2}\Lambda_0^{(d-3-f)} + S_{d-3}\Lambda_1^{(d-3-f)} + \ldots S_0\Lambda_{d-2}^{(d-3-f)} &= 0. \end{aligned}$$

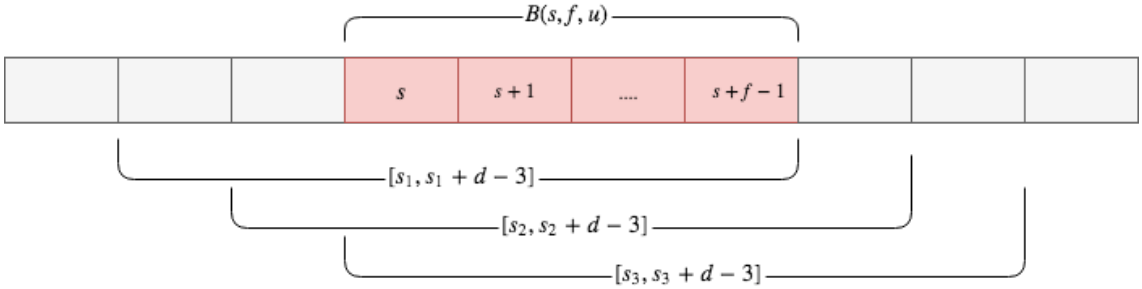

Figure 4.2: Graphical illustration of Theorem 4.2.

Define

$$\bar{\Lambda}^{(p)}(x) = \prod_{i=-(p-1)}^{0}(1 - a^i x)$$

where p is related to the maximum allowable length of burst, in this case $p = d - 2$. Also define

$$\Gamma(x) = S_{d-2}\bar{\Lambda}_0^{(d-2)} + S_{d-3}\bar{\Lambda}_1^{(d-2)}x + \cdots + S_0\bar{\Lambda}_{d-2}^{(d-2)}x^{d-2}$$

Now, note that

$$S_{d-2}\Lambda_0^{(i)} + S_{d-3}\Lambda_1^{(i)} + \cdots + S_0\Lambda_{d-2}^{(i)} =$$

$$S_{d-2}\bar{\Lambda}_0^{(d-2)} + S_{d-3}\bar{\Lambda}_1^{(d-2)}a^{s-i+d-3} + \cdots + S_0\bar{\Lambda}_{d-2}^{(d-2)}a^{(d-2)(s-i+d-3)} =$$

$$\Gamma(a^{s-i+d-3})$$

By the last relation, *Theorem 4.2* is translated to the following theorem.

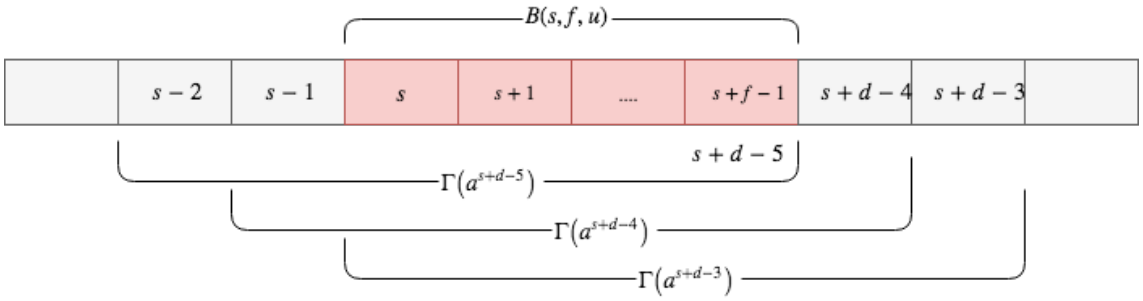

Figure 4.3: Graphical illustration of Theorem 4.3.

Theorem 4.3. *A single burst* $\mathsf{B}(s, f, \mathbf{u})$ *has occurred if and only if* $\Gamma(x)$ *has* $d - f - 1$ *consecutive roots* $a^{s+f-1}, a^{s+f}, \ldots, a^{s+d-3}$.

According to Theorem 4.3 the first index of the consecutive root sequence of $\Gamma(x)$ corresponds to the end of the error burst, as it is illustrated on Figure 4.3. For this reason it is easier to keep track of the end of the error burst instead of it's start. Also note that $\Gamma(x)$ has degree up to $d-2$ and thus it may have up to $d-2$ valid roots. That is up to $d-2$ candidate error bursts in the worst case scenario. The objective is to determine the shortest error burst, which is given by the longest consecutive root sequence of $\Gamma(x)$. Finally given a burst $\mathsf{B}(s, f, \mathbf{u})$, the error locator polynomial is calculated as $\Lambda(x) = \bar{\Lambda}^{(f)}(a^{s+f-1})$, which can be later used in Forney's formula to calculate the error magnitudes.

All of the above can be summarized into the algorithmic procedure of *Algorithm 3*.

Algorithm 3: Reed-Solomon burst error decoding algorithm

1. Precalculate the coefficients of $\bar{\Lambda}^{(d-2)}(x)$
2. Run *Algorithm 2*, if successful then stop.
3. Compute the coefficients of $\Gamma(x)$.
4. Apply the chien search on $\Gamma(x)$ to obtain the shortest single-burst by tracking the longest consecutive root sequence. Record the end position of the desired shortest burst.
5. Compute the burst error magnitudes using Forney's formula.
6. Correct the received polynomial and retrieve the message from the last k positions.
7. Return $m(x)$

As an example of the above, consider the (24,16) shortened Reed-Solomon code over GF(2^5). The design distance of the code is $d = 9$, thus, the burst error correction capability is $f = d - 2 = 7$. Let the transmitted codeword be
$\mathbf{c} = [a^8, a^3, a^{21}, a^8, a^{21}, a^0, a^3, a^{18}, a^1, a^{27}, a^1, a^{25}, a^{12}, a^{26}, a^{10}, a^{10}, a^{17}, a^{28}, a^2, a^{29}, a^{18}, a^3, a^8, a^{22}]$
and the received word
$\mathbf{r} = [a^8, \underline{a^{26}, a^1, a^2, a^{17}, a^{14}, a^{29}}, a^{18}, a^1, a^{27}, a^1, a^{25}, a^{12}, a^{26}, a^{10}, a^{10}, a^{17}, a^{28}, a^2, a^{29}, a^{18}, a^3, a^8, a^{22}]$
Its syndrome values are computed as
$\mathbf{S} = [a^{28}, a^{12}, a^4, a^4, a^{27}, a^{13}, a^{28}, a^{17}]$,
which yield

$\Gamma(x) = a^7x^7 + a^{26}x^6 + a^{10}x^5 + a^{16}x^4 + a^{11}x^3 + a^{28}x^2 + a^{26}x^1 + a^{17}$.

The chien search identifies five valid roots, $a^6, a^7, a^9, a^{16}, a^{22}$, respectively. By *Theorem 4.3*, the first two consecutive roots result in a burst of length 6, which leads to the candidate codeword

$\mathbf{c_1} = [a^8, \underline{a^3, a^{21}, a^8, a^{21}, a^0, a^3}, a^{18}, a^1, a^{27}, a^1, a^{25}, a^{12}, a^{26}, a^{10}, a^{10}, a^{17}, a^{28}, a^2, a^{29}, a^{18}, a^3, a^8, a^{22}]$.

The root a^9 leads to a candidate codeword

$\mathbf{c_2} = [a^8, a^{26}, a^1, \underline{a^{27}, a^{15}, a^3, a^{20}, a^0, a^{24}, a^9}, a^1, a^{25}, a^{12}, a^{26}, a^{10}, a^{10}, a^{17}, a^{28}, a^2, a^{29}, a^{18}, a^3, a^8, a^{22}]$.

The root a^{16} leads to a candidate codeword:

$\mathbf{c_3} = [a^8, a^{26}, a^1, a^2, a^{17}, a^{14}, a^{29}, a^{18}, a^1, a^{27}, \underline{a^{15}, a^9, 0, a^5, a^3, a^9, a^{30}}, a^{28}, a^2, a^{29}, a^{18}, a^3, a^8, a^{22}]$.

The last root, a^{22} results in a candidate codeword:

$\mathbf{c_4} = [a^8, a^{26}, a^1, a^2, a^{17}, a^{14}, a^{29}, a^{18}, a^1, a^{27}, a^1, a^{25}, a^{12}, a^{26}, a^{10}, a^{10}, \underline{a^2, a^{23}, a^{21}, a^{17}, a^{15}, a^{28}, a^{15}}, a^{22}]$.

The algorithm picks the shortest burst, which corresponds to the longest consecutive root sequence thereby successfully retrieves the transmitted codeword. The last example is illustrated in the figures below.

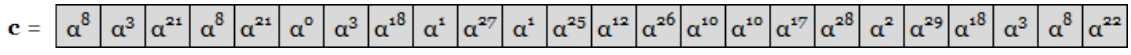

Figure 4.4: Transmitted codeword.

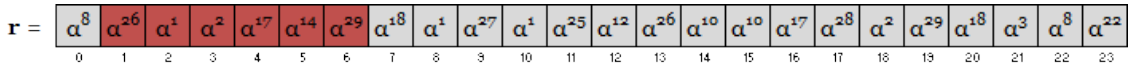

Figure 4.5: Received word.

Figure 4.6: Candidate codeword of the consecutive roots a^6, a^7.

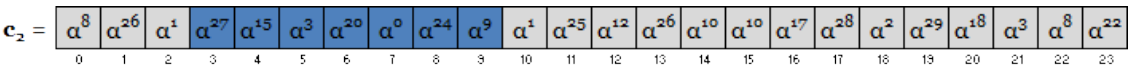

Figure 4.7: Candidate codeword of the consecutive roots a^9.

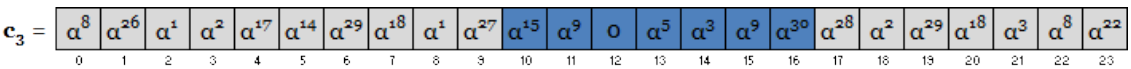

Figure 4.8: Candidate codeword of the consecutive roots a^{16}.

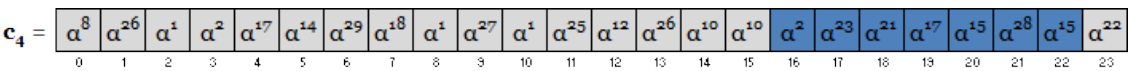

Figure 4.9: Candidate codeword of the consecutive roots a^{22}.

Figure 4.10 presents the results of a simulation of an $(255, 237)$ RS code over GF(256). The error correction capability of the code is $t = 9$ and the design distance is $d = 19$. During the simulation we generate random codewords, and then we add a random burst

error of length f between $t = 9$ and $d - 2$, which can be seen on the horizontal axis. Trying to correct the burst error, we keep track of the probability of decoding failure of the conventional decoding algorithm in comparison with the probability of decoding failure of the proposed algorithm. The decoding failure probability is displayed on the vertical axis. On the same figure we present the theoretical upper bound of the decoding failure probability of the proposed decoding algorithm.

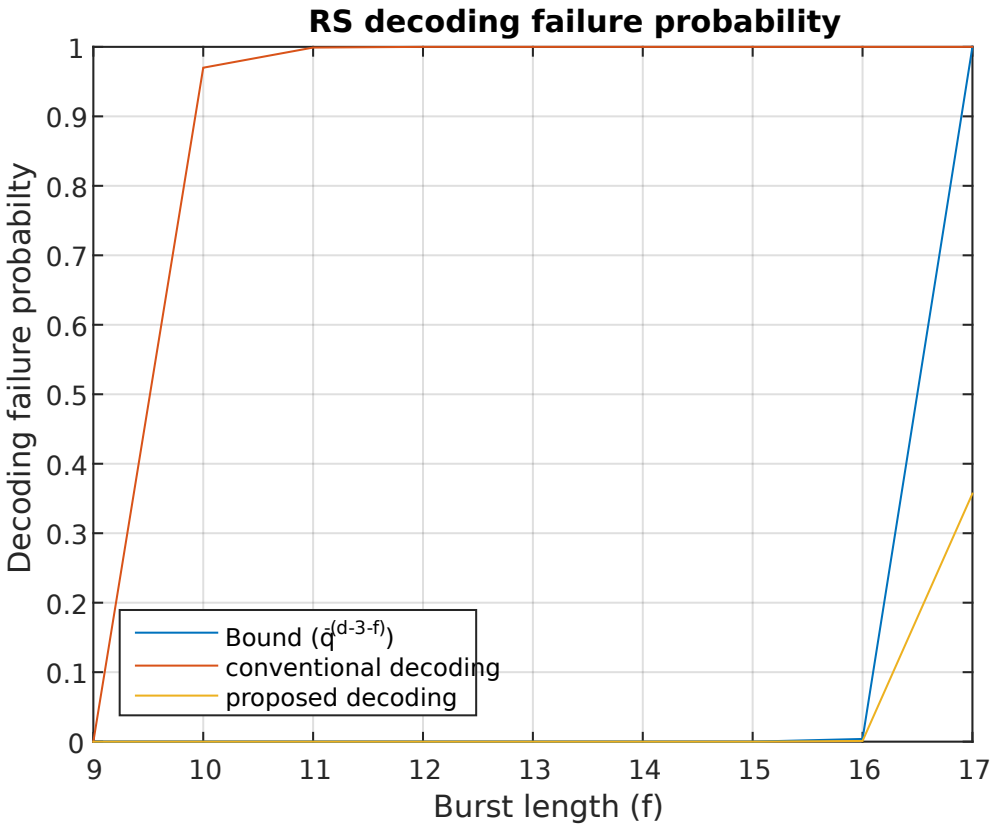

Figure 4.10: RS decoding simulation.

Observe that the conventional decoding corrects all burst errors with length f up to $t = 9$ with probability 1. For burst errors of larger length, the conventional decoding fails with a very high probability. Whereas the proposed algorithm starts failing when the burst error length reaches the burst error length limit of $f = d-2$. Note that, if f exceeds $d - 2$ then both decoding techniques fail with probability 1.

Appendices

Appendix A

A.1 Theorem 1.4

Proof. For any a, b, c in $\mathbb{Z}_n$ it holds that:

- $a \oplus b = a + b \pmod{n} = r$ where r is an integer with $0 \leq r \leq n-1$ and therefore $r \in \mathbb{Z}_n$. Thus $\mathbb{Z}_n$ is closed under addition modulo q.

- In order to show that the associative property holds, i.e.,
  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$, we start by showing that:
  $(a \oplus b) \oplus c = (a + b + c) \pmod{n}$.
  By the division algorithm, it holds that:
  $(a + b) = d_1 n + r_1 \implies r_1 = (a + b) - d_1 n$, with $0 \leq r_1 < n$
  Then, $(a \oplus b) \oplus c = (((a+b) \pmod{n}) + c) \pmod{n} = r_1 + c \pmod{n} = (a+b) - d_1 n + c \pmod{n}$
  By employing the division algorithm again we get that:
  $(a + b) - d_1 n + c = d_2 n + r_2 \implies r_2 = a + b + c - (d_1 + d_2)n$, with $0 \leq r_2 < n$
  So it holds that $(a \oplus b) \oplus c = r_2$
  By the division algorithm on $(a + b + c) \pmod{n} = r_3$ we get that:
  $a + b + c = d_3 n + r_3 \implies r_3 = a + b + c - d_3 n$, with $0 \leq r_3 < n$.
  Now note that from $0 \leq r_2 < n$ and $0 \leq r_3 < n$ it holds that:
  $-n < r_2 - r_3 < n \implies$
  $-n < (a + b + c - (d_1 + d_2)n) - ((a + b + c) - d_3 n) < n \implies$
  $-n < (d_3 - d_1 - d_2)n < n \implies$
  $-1 < d_3 - d_1 - d_3 < 1$, but $d_3 - d_1 - d_3$ is an integer (as sum of integers) and so it must be the case that $d_3 - d_1 - d_3 = 0 \implies d_3 = d_1 + d_2$.
  Finally we find that $(a \oplus b) \oplus c = r_2 = a + b + c - (d_1 + d_2)n = a + b + c - (d_3)n = r_3 = (a + b + c) \pmod{n}$. Therefore it holds that:

  $$(a \oplus b) \oplus c = (a + b + c) \pmod{n}.$$

  In a similar manner we can also show that $(a \oplus (b \oplus c)) = (a + b + c) \pmod{n}$ and thus proving that associativity holds.

- The element 0 acts as the identity element since:
  $a \oplus 0 = 0 \oplus a = a + 0 \pmod{n} = a \pmod{n} = a$.

- If $a \neq 0$, let $b = n - a \implies 1 \leq b \leq n - 1 \implies b \in \mathbb{Z}_q$. Now note that $a \oplus b = a + b \pmod{n} = a + n - a \pmod{n} = n \pmod{n} = 0$ thus b is the inverse of a. There is also the case that $a = 0$ in which, its inverse is the element 0. Thus every element in $\mathbb{Z}_n$ has an inverse.

- Commutativity also holds, since:
  $a \oplus b = a + b \pmod{n} = b + a \pmod{n} = b \oplus a$.

Thus $< \mathbb{Z}_n, \oplus >$ is an Abelian group. □

A.2 Theorem 1.5

Proof. For any a, b, c in $\mathbb{Z}_n$ it holds that:

- $a * b = a \cdot b \pmod{n} = r$, with $0 \leq r \leq n-1 \implies r \in \mathbb{Z}_n$ thus $\mathbb{Z}_n$ is closed under multiplication. The same holds for addition.

- According to *Theoreom 1.4*, $< \mathbb{Z}_n, \oplus >$ forms a commutative (Abelian) group. The additive identity is 0.

- We will now show that multiplication is associative.
  In order to do so, we will begin by showing that:

  $$(a * b) * c = a \cdot b \cdot c \pmod{n}.$$

  We can write $(a * b) * c = (((a \cdot b) \pmod{n}) \cdot c) \pmod{n}$
  By utilizing the division algorithm we get:
  $(a \cdot b) \pmod{n}) = r_1$, with $a \cdot b = d_1 \cdot n + r_1 \implies r_1 = a \cdot b - d_1 \cdot n$ and $0 \leq r_1 < n$.
  Thus now we can write

  $$(a * b) * c = (((a \cdot b) \pmod{n}) \cdot c) \pmod{n} = (r_1 \cdot c) \pmod{n}.$$

  Again by the division algorithm we get:
  $(r_1 \cdot c) \pmod{n} = r_2$ with $r_1 \cdot c = d_2 \cdot n + r_2 \implies r_2 = r_1 \cdot c - d_2 \cdot n \implies r_2 = (a \cdot b - d_1 \cdot n) \cdot c - d_2 \cdot n \implies r_2 = a \cdot b \cdot c - (d_1 \cdot c + d_2) \cdot n$, where $0 \leq r_2 < n$.
  Finally by the division algorithm on $a \cdot b \cdot c$ we can write:
  $a \cdot b \cdot c \pmod{n} = r_3$ where $a \cdot b \cdot c = d_3 \cdot n + r_3 \implies r_3 = a \cdot b \cdot c - d_3 \cdot n$, where $0 \leq r_3 < n$.
  Now note that from: $0 \leq r_2 < n$ and $0 \leq r_3 < n$, it holds that:
  $-n < r_2 - r_3 < n \implies$
  $-n < (a \cdot b \cdot c - (d_1 \cdot c + d_2) \cdot n) - (a \cdot b \cdot c - d_3 \cdot n) < n \implies$
  $-n < (d_3 - d_1 \cdot c - d_2) \cdot n < n \implies -1 < d_3 - d_1 \cdot c - d_2 < 1$, but $d_3 - d_1 \cdot c - d_2$ is an integer so it must be the case that:
  $d_3 - d_1 \cdot c - d_2 = 0 \implies d_3 = d_1 \cdot c + d_2$.
  Thus it holds that:
  $(a * b) * c = r_2 = a \cdot b \cdot c - (d_1 \cdot c + d_2) = a \cdot b \cdot c - d_3 \cdot n = r_3 =$
  $= (a \cdot b \cdot c) \pmod{n}$.
  In a similar manner we can show that $a * (b * c) = (a \cdot b \cdot c) \pmod{n}$ and thus proving that associativity holds.

- The left distributivity law holds: $a * (b \oplus c) = (a * (b + c) \pmod n) = (a \cdot (b + c)) \pmod n = (a \cdot b + a \cdot c) \pmod n =$
  $((a \cdot b) \pmod n + (a \cdot c) \pmod n) \pmod n = (a * b + a * c) \pmod n = (a * b) \oplus (a * c)$.
  Similarly it can be shown that the right distributivity law holds:
  $(a \oplus b) * c = (a * c) \oplus (b * c)$.

- Multiplication is commutative: $a * b = (a \cdot b) \pmod n = (b \cdot a) \pmod n = b * a$.

- The multiplicative identity is the element 1: $(a * 1) = (1 * a) = (1 \cdot a) \pmod n = a \pmod n = a$.

Thus $< \mathbb{Z}_n, \oplus, * >$ forms a commutative ring with identity. □

References

[1] D. Burton, *Elementary Number Theory*, ser. Asia Higher Education Mathematics and Statistics Higher Mathematics. McGraw-Hill, 2010.

[2] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. New York, NY, USA: Wiley-Interscience, 2005.

[3] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[4] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A method for solving key equation for decoding goppa codes," *Information and Control*, vol. 27, no. 1, pp. 87 – 99, 1975.

[5] J. Massey, "Shift-register synthesis and bch decoding," *IEEE Transactions on Information Theory*, vol. 15, no. 1, pp. 122–127, Jan 1969.

[6] Y. Wu, "Novel burst error correction algorithms for reed-solomon codes," *IEEE Transactions on Information Theory*, vol. 58, no. 2, pp. 519–529, Feb 2012.