Reed-Solomon Encoding and Decoding

A Visual Representation

León van de Pavert

30 June 2011

Abstract

Bachelor's Thesis
Turku University of Applied Sciences
Degree Programme in Information Technology
Spring 2011 | 21 pages
Author: León van de Pavert
Instructor: Hazem Al-Bermanei

VISUAL REPRESENTATION OF BURST ERRORS AND REED-SOLOMON DECODING

The capacity of a binary channel is increased by adding extra bits to this data. This improves the quality of digital data. The process of adding redundant bits is known as channel encoding.

In many situations, errors, are not distributed at random but occur in bursts. For example, scratches, dust or fingerprints on a compact disc (CD) introduce errors on neighbouring data bits. Cross interleaved Reed-Solomon Codes (CIRC) are particular well-suited for detection and correction of burst errors and erasures. Interleaving redistributes the data over many blocks of code. The double encoding has the first code declaring the erasures. The second code corrects them.

The purpose of this thesis is to present Reed-Solomon error correction codes in relation to burst errors. In particular, this thesis visualises the mechanism of cross-interleaving and its ability to allow for detection and correction of burst errors.

KEYWORDS: Coding theory, Reed-Solomon code, burst errors, cross-interleaving, compact disc

Acknowledgements

It is a pleasure to thank those who supported me making this thesis possible.

I am thankful to my supervisor, Hazem Al-Bermanei, whose intricate knowledge of coding theory inspired me, and whose lectures, encouragement, and support enabled me to develop an understanding of this subject.

This thesis would not have been possible without the support of the teachers at the University of Applied Sciences and I would not have been able to even start these studies without the support and understanding of my wife, Maija and motivation from my children Kira, Caspar and Julius.

Last but certainly not least, I would like to express my gratitude to Aschwin van der Woude, for listening to my issues, for his advice and coaching.

Contents

List of Figures

List of Tables

List of Symbols and Notations

Polynomials have their respective degree shown in brackets.

Variables

n	length of codeword	
k	number of data symbols	
d	distance	
d_{min}	minimum distance	
t	number of correctable errors	
l	number of detectable errors	

Polynomials

$g(x)$	generator	$(n-k)$
$p(x)$	error check	$(n-k-1)$
$h(x)$	parity check	(k)
$i(x)$	information	$(k-1)$
$c(x)$	code-word	$(n-1)$
$c'(x)$	received code-word	$(n-1)$
$c'_r(x)$	corrected code-word	$(n-1)$
$s(x)$	syndrome	$(n-k-1)$
$e(x)$	error	$(n-1)$

Number Sets

$\mathbb{N}$	set of natural numbers {0, 1, 2, ...}
$GF(q)$	Galois field or finite field where $q \in \mathbb{N}$

Chapter 1

Introduction

1.1 Error detection and correction

When data is stored or transmitted, we cannot ignore encoding. The field of mathematics that deals with sending data, a digital bit stream, over a noisy channel is called coding theory. The Oxford English Dictionary says the following about code:

> Any system of symbols and rules for expressing information or instructions in a form usable by a computer or other machine for processing or transmitting information.

During World War II, and even before, as far back as classic times, messages had to be send to allies but it was crucial they were unintelligible to the enemy. This field of cryptology was born out of necessity, a sense of survival. After the war, before governments could render the research obsolete, the people behind cryptology research showed that cryptology and eventually the theory of error detecting and correcting could be put into practical use. We can see that the field of cryptology is adjacent to and often-times overlapping with the field of coding theory (Trappe & Washington, 2006).

This thesis will briefly look at some pioneers and their achievements, then goes through the basics of coding theory and addresses the mathematics behind coding in chapter 2. While chapter 3 goes into the theory of linear block codes, it will be the visualisation in chapter 4 that explains how burst errors can be detected and corrected, on e.g. a Compact Disc. Physical damage like dust or scratches or material impurities can cause erasures or burst errors in the data stream. With forward error correction techniques like Reed-Solomon codes these interrupts in the data stream can be detected and corrected.

1.2 History of Error Control Coding

Pioneers of coding theory are Shannon, Hamming who were colleagues at Bell Labs. Hocquenghem in 1959 and independently Bose and Ray-Chaudhuri in 1960 were responsible for a class of codes known an BCH codes. Reed and Solomon followed with a set of cyclic codes, which are BCH codes, but are well suited for detecting and correcting burst errors and erasures. It wasn't until almost a decade later when Berlekamp invented a decoding algorithm which was simplified by Massey in 1969. In 1982 the Compact Disc (CD) was the first mass-produced device that used these error correcting codes.

1.2.1 Shannon

Claude Shannon (1916–2001) was an electronic engineer and mathematician and during the Second World War he joined Bell Labs to work on cryptography. His work was closely related to coding theory and eventually led to publication of the article named A Mathematical Theory of Communication in 1948, which is now regarded as one of the founding works of communication theory. Presently, not only do many regard him as the father of information theory, but he is also credited with establishing digital computer and digital circuit design theory while he was a master's student at MIT (Bose, 2008).

1.2.2 Hamming

Richard Hamming (1915–1998) was a contemporary and colleague of Shannon at Bell Labs. While doing research on cryptology, Hamming became interested in the idea of error correcting codes while working on a relay computer out of normal office hours. Unfortunately there were no computer operators available to react to an alarm in case an error was detected. Hamming had to devise a code that would not only detect an error, but would also be able to correct it automatically, instead of just ringing the alarm. These codes are used to add redundancy to data which aid the detection and correction of errors. Chapter 3 explains the Hamming code, which was a first in the the field we now know as coding theory.

Although the Hamming code was referred to by Shannon in 1948, patent considerations prevented its independent publication until 1950.

1.2.3 Hocquenghem, Bose and Ray-Chaudhuri

Alexis Hocquenghem (1908?–1990) was a French mathematician, whose article "Codes correcteurs d'erreurs" from 1959 mentioned codes that he described as a "generalization of Hamming's work" (Hocquenghem, 1959).

Independently from Hocquenghem, Ph.D. adviser Raj Bose (1901–1987) and his student Dwijendra Ray-Chaudhuri (1933–) published "On a class of error correcting binary group codes" in 1960. This class of linear block codes is named after Bose, Ray-Chaudhuri and Hocquenghem and became known as BCH codes (Wicker & Bhargava, 1999).

1.2.4 Reed and Solomon

Irving Reed (1923–) is an American mathematician and engineer who is best known for co-inventing a class of algebraic codes known as Reed-Solomon (RS) codes in collaboration with Gustave Solomon (1930–1996). RS codes are seen as a special case of the larger class of BCH codes but it wasn't until almost a decade later, by regarding them as cyclic BCH codes, that an efficient decoding algorithm gave them the potential to their widespread application.

1.2.5 Berlekamp and Massey

Elwyn Berlekamp (1940–) is a professor emeritus of mathematics, electrical engineering and computer science at the University of California, Berkely. While he was studying electrical engineering at MIT one of his Ph.D. advisers was Claude Shannon. Berlekamp invented an algorithm for decoding BCH codes in 1968, but it was James Massey (1934–), an information theorist and cryptographer, who simplified this algorithm in 1968 which we know as the Berlekamp-Massey algorithm (Massey, 1969).

This algorythm made it possible to develop a fast and efficient decoder with a linear feedback shift register (LSFR), but it was not until 1982 with the advent of the mass production of the Compact Disc that the digital information age as we know it was started. Today RS codes are widely in use in many applications that involve data transmission, like computer networks: TCP/IP; telephony: GSM, GPRS, UMTS; digital video broadcasting: DVB, and data storage, like Hard-drives in computers; memory cards in cameras and telephones and optical storage like Compact Discs (CD), Digital Versitile Discs (DVD) and Blu-ray Discs (BD).

1.3 Basics of Data Communication

A sender transmits a message through a channel to a receiver. The channel in these cases could be air when usin a wireless network or an ethernet cable. Noise may appear on these channels, so in order to receive the message with as few errors as possible, idealy the sender should use high power signal amplification and the channel should be as short as possible. However, in normal situations however these are not viable solutions. GSM telephones have in fact very small batteries and are rather energy efficient and the ethernet cable in a building can be up to 100 meters before an active repeater or switch has to amplify the signal. In order to use as little energy as possible and transmit over a long distance, codewords have to be encoded, as shown in Figure 1.1. It is then transmitted over a channel where errors may be introduced. The received codeword needs to be decoded into the received message.

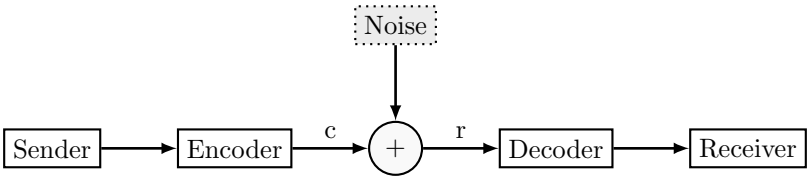

Figure 1.1: Digital transmission system

The probability that codeword r is received if code-word c is transmitted can be expressed as $P(r|c)$. In Figure 1.2, a model of a binary symmetric channel (BSC) shows the event that a 1 is transmitted. There is a probability that 0 is received. The transmission is unaltered with a probability of $1 - p$ (Bossert, 1999).

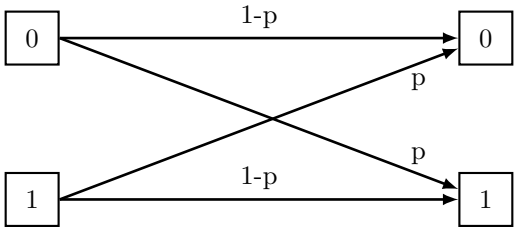

Figure 1.2: Model of the binary symmetric channel (BSC)

Chapter 2

Coding Theory Basics

2.1 Linear Algebra

Mathematicians developed their coding theory using linear algebra, which works with set of numbers or fields. These numbers can be added, subtracted, multiplied or divided. Fields, like integers, or the set of natural numbers $\mathbb{N} = \{0, 1, 2, ...\}$ is an infinite field; we could always imagine its largest element and add 1 to it.

Information and code can be seen as elements in a finite field which comes with some advantages when using the binary number system.

2.2 Galois Fields

A finite field F_q is a field F which has a finite numer of elements and q is the order of the field. This finite field is often called a Galois field, after the French mathematician Évariste Galois (1811 – 1832) and is denoted $GF(q)$. For the purpose this thesis we consider only binary field $GF(2)$ and its extension fields $GF(2^m)$ where $m \in \{2, 3, 4, ...\}$.

The following is always valid for all numbers in a binary Galois field (Blahut, 1983):

- Fields contain 0 or 1.
- Adding two numbers gives one number in the set.
- Subtracting two numbers gives one number in the set.
- Multiplying one number gives one number in the set.
- Dividing one number by 1, as division by 0 is not allowed, gives one number in the set.
- The distributive law, $(a + b)c = ac + bc$, holds for all elements in the field.

A finite field, by definition, has to contain at least two numbers and therefore the smallest Galois field contains the elements or numbers 0 and 1, and is defined as $GF(2) = \{0, 1\}$. Since we have a finite field with only aforementioned binary numbers, the addition of 1 and 1 in Table 2.1 cannot be equal to 2, but instead has to be defined as 1+1=0 where 2 is congruent to 0 modulo 2, or $2 \equiv 0 \pmod 2$ (Hill, 1986). For subtraction we take as the additive inverse of $-a$. This inverse can be found by $a + b = c$ and write it $b = c - a$ which is equal to $b = c + (-a)$. Substituting a

Table 2.1: Addition for $GF(2)$.

+	**0**	**1**
0	0	1
1	1	0

Table 2.2: Multiplication for $GF(2)$.

*	**0**	**1**
0	0	0
1	0	1

and b with 0 and 1, we can see that the additive inverse of 0 is 0 and the additive inverse of 1 is 1.

Division is a multiplication with its multiplicative inverse of which we can write as:

$$\frac{a}{b} = c$$

Therefore $a \cdot b^{-1} = c$ which results in $a = c \cdot b$. Because $a \cdot a^{-1} = 1$, the multiplicative inverse of 1 is 1. Division is always possible for all except 0. Because division by zero is not defined and $0 \cdot a^{-1} \neq 1$, zero has no multiplicative inverse.

2.3 Extension Fields

Finite fields exist for all prime numbers q and for all q^m where q is prime and m is a positive integer. $GF(q)$ is a sub-field of $GF(q^m)$ and as such the elements of $GF(q)$ are a sub-set of the elements of $GF(q^m)$, therefore $GF(q^m)$ is an extension field of $GF(q)$.

Table 2.3: Addition for $GF(4) = \{0, 1, 2, 3\}$

+	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

Table 2.4: Multiplication for $GF(4) = \{0, 1, 2, 3\}$

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

Consider $GF(4) = \{0, 1, 2, 3\}$ in Table 2.3 and Table 2.4, which is not a Galois field because it is of order 4, which is not a prime. The element 2 has no multiplicative inverse and therefore we cannot divide by 2. Instead, we could define $GF(4) = \{0, 1, a, b\}$ with addition and multiplication as shown in Table 2.5 and Table 2.6. Now all elements do have additive and multiplicative inverses.

Table 2.5: Addition for $GF(4) = \{0, 1, a, b\}$

+	**0**	**1**	**a**	**b**
0	0	1	a	b
1	1	0	b	a
a	a	b	0	1
b	b	a	1	0

Table 2.6: Multiplication for $GF(4) = \{0, 1, a, b\}$

*	**0**	**1**	**a**	**b**
0	0	0	0	0
1	0	1	a	b
a	0	a	b	1
b	0	b	1	a

These extension fields are used to handle non-binary codes where code symbols are expressed as m-bit binary code symbols, For example, $GF(4)$ consists of four different two-bit symbols and $GF(16)$ of 16 hexadecimal symbols. To obtain multiplication for binary numbers are expressed

as polynomials, they are multiplied and divided by the prime polynomial while the remainder is taken as result.

2.4 Polynomials

Let us we write $GF(4)$ as $GF(2^2)$ and take prime polynomial

$$p(x) = x^2 + x + 1$$

which is an irreducible polynomial of degree 2, which can be checked by multiplying $p(x)$ with polynomials of a lesser degree, like 1, x and $x + 1$ (Blahut, 1983).

Table 2.7: Addition for $GF(2^2)$ in binary representation.

+	**00**	**01**	**10**	**11**
00	00	01	10	11
01	01	00	11	01
10	10	11	00	01
11	11	10	01	00

Table 2.8: Multiplication for $GF(2^2)$ in binary representation.

*	**00**	**01**	**10**	**11**
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

This gives us the structure of $GF(2^2)$ in Table 2.7 and Table 2.8. Note that addition in a finite field is equivalent to the logic exclusive OR (XOR) operation and multiplication is equivalent to the logic AND. In Table 2.9 and Table 2.10, $GF(2^2)$ is represented in polynomial form.

Table 2.9: Addition for $GF(2^2)$ in polynomial representation.

+	**0**	**1**	**x**	**x+1**
0	0	1	x	x+1
1	1	0	x+1	x
x	x	x+1	0	1
x+1	x+1	x	1	0

Table 2.10: Multiplication for $GF(2^2)$ in polynomial representation.

+	**0**	**1**	**x**	**x+1**
0	0	0	0	0
1	0	1	x	x+1
x	0	x	x+1	1
x+1	0	x+1	1	x

In order to describe an extension field $GF(q^m)$ it is useful to know its a primitive polynomial $p(x)$, where the degree of $p(x)$ is equal to m. For example,

$$GF(16) = GF(2^4) = \{0000, 0001, 0010, ..., 1111\}$$

is a finite field that contains 16 4-bit code symbols. Addition is analogue to the example above. Multiplication can be obtained firstly by writing the symbols as polynomials to express which positions in these 4-bit codes are non-zero and secondly by using modulo 2 addition of coefficients in addition and multiplication.
Let α be defined as a root of polynomial $p(x)$, such that we can write:

$$p(\alpha) = 0$$

Thus for $GF(16)$ with its irreducible polynomial $p(x) = x^4 + x + 1$ we can write:

$$\alpha^4 + \alpha + 1 = 0$$
$$\alpha^4 = 0 - \alpha - 1$$

We have already noted that subtraction is the same as addition in a binary finite field, so:

$$\alpha^4 = \alpha + 1$$

Therefore the polynomial of exponential α^4 is $\alpha+1$. From there we can calculate the polynomial for α^5 by:

$$\begin{aligned}\alpha^5 &= \alpha \cdot \alpha^4 \\ &= \alpha \cdot (\alpha + 1) \\ &= \alpha^2 + \alpha\end{aligned}$$

Now we can take $\alpha^k = \alpha \cdot \alpha^{k-1}$ for every $k < 2^m - 1$, where $m = 4$ in our example. Calculations for α^5 and α^6 in Table 2.11 are straight forward, however, polynomials of degree 4 may be reduced to ones of less than a degree of 4:

$$\begin{aligned}\alpha^7 &= \alpha \cdot \alpha^6 \\ &= \alpha \cdot (\alpha^3 + \alpha^2) \\ &= \alpha^4 + \alpha^3\end{aligned}$$

Substituting α^4 with $\alpha + 1$ gives

$$\begin{aligned}\alpha^7 &= \alpha + 1 + \alpha^3 \\ &= \alpha^3 + \alpha + 1\end{aligned}$$

so the polynomial of α^6 is $x^3 + x + 1$. The remaining exponentials can be obtained in the same manner while keeping each polynomial of degree 3 or less because we can substitute α^4, a polynomial of degree 4, with $\alpha + 1$, which is of degree 1. Note that $\alpha^{15} = 1$. Fermats's Little Theorem says that $q^{m-1} \equiv 1(mod\ m)$, where q is prime and m is a positive integer (Blahut, 1983) (Bossert, 1999).

Table 2.11: Elements of Galois Field $GF(2^4)$ in different notations.

Exponential	**Algebraic**	**Polynomial**	**Bin**	**Dec**	**Hex**
0	0	0	0000	0	0
α^0	1	1	0001	1	1
α^1	α	x	0010	2	2
α^2	α^2	x^2	0100	4	4
α^3	α^3	x^3	1000	8	8
α^4	$\alpha + 1$	$x + 1$	0011	3	3
α^5	$\alpha(\alpha + 1)$	$x^2 + x$	0110	6	6
α^6	$\alpha(\alpha^2 + \alpha)$	$x^3 + x^2$	1100	12	C

Continuation of Table 2.11

Exponential	Algebraic	Polynomial	Bin	Dec	Hex
α^7	$\alpha(\alpha^3+\alpha^2)$	x^3+x+1	1011	11	B
α^8	$\alpha(\alpha^3+\alpha+1)$	x^2+1	0101	5	5
α^9	$\alpha(\alpha^2+1)$	x^3+x	1010	10	A
α^{10}	$\alpha(\alpha^3+\alpha)$	x^2+x+1	0111	7	7
α^{11}	$\alpha(\alpha^2+\alpha+1)$	x^3+x^2+x	1110	14	E
α^{12}	$\alpha(\alpha^3+\alpha^2+\alpha)$	x^3+x^2+x+1	1111	15	F
α^{13}	$\alpha(\alpha^3+\alpha^2+\alpha+1)$	x^3+x^2+1	1101	13	D
α^{14}	$\alpha(\alpha^3+\alpha^2+1)$	x^3+1	1001	9	9

2.5 Vector Space

Linear codes can be represented as sets of vectors. Let us define a vector space $GF(q^m)$. This is a vector space of a finite dimension, m. The codewords are q-ary sets of m-elements or m-tuples which form the coordinates of the endpoints of the vectors. Figure 2.1 presents two of such m-dimensional vector spaces. In such a vector space every codeword can be presented as a the sum of two vectors give another vector in the same vector space (Bose, 2008). For example, $GF(2^2)$ is a two-dimensional vector space. It has four binary vectors. Take vectors $v_1 = [0,1]$, $v_2 = [1,0]$ and $v_3 = [1,1]$, then $v_1 + v_2 = [0,1] + [1,0] = [1,1]$, which is a vector in the same space.

Vectors $v_1, v_2, ..., v_k$ are linear independent if there is not a single set of scalars $a_i \neq 0$, such that $a_1v_1 + a_2v_2 + ... + a_kv_k = 0$. For example, vectors $[0,1]$ and $[1,0]$ are linearly independent, but $[0,1]$ and $[1,1]$ are linear dependent vectors.

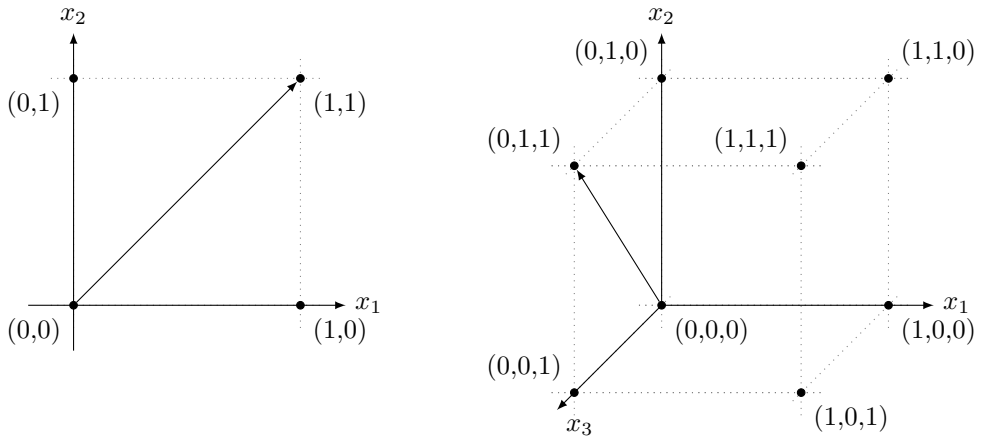

Figure 2.1: Codewords $[1,1]$ and $[0,1,1]$ as vectors over $GF(2)$

Chapter 3

Linear Block Codes

3.1 Hamming weight, Minimum Distance and Code Rate

The Hamming weight $w_H(x)$ of a codeword or vector x is defined as the amount of non-zero elements or vector coordinates, which ranges from zero to length n of said codeword.

$$w_H(x) = \sum_{j=0}^{n-1} w_H(x_j), \text{ where } w_H(x_j) = \begin{cases} 0, & \text{if } x_j = 0 \\ 1, & \text{if } x_j \neq 0 \end{cases}$$

The Hamming distance $d_H(x, y)$ between two codewords or vectors x and y is defined as amount of elements or coordinates where x and y differ.

$$d_H(x, y) = \sum_{j=0}^{n-1} w_H(x_j + y_j), \text{ where } w_H(x_j + y_j) = \begin{cases} 0, \text{if } x_j = y_j \\ 1, \text{if } x_j \neq y_j \end{cases}$$

$$d_H(x, y) = w_H(x, y)$$

The minimum distance $d_m in$ of code C is the minimum distance between two different codewords. The minimum distance for linear codes is equal to the minimum weight (Bossert, 1999). However, a codeword containing only zeros and therefore having a distance of zero is disregarded as the minimum distance cannot be zero.

Let x, y be codewords in code C. A received vector, which is the sent vector x in C, plus error vector e can only be corrected if the distance between any other codeword y in C fulfill

$$d_{min}(x, x + e) < d_{min}(y, x + e) \text{ or } w_{min}(e) < w_{min}(x + y + e).$$

Therefore

$$w_{min}(e) \leq \frac{d - 1}{2},$$

where d is the distance. This is written as

$$t \leq \frac{d - 1}{2} \text{ or } d \geq 2t + 1,$$

where t is the amount of errors that can be corrected.

In general, a code C of length n, with M codewords, and a minimum distance $d = d(C)$, is called an (n, M, d) code. Then M$\leq q^{n-d+1}$ and the coderate of a q-ary (n, M, d) code is at most $1 - \frac{d-1}{n}$.

A linear q-ary code of lenght n, with k codewords or message symbols, and distance d is called a (n, k, d) code or (n, k) code. The code rate is defined as

$$R = \frac{log_q k}{n}.$$

If, according to Shannon's channel coding theorem, rate R is less than capacity C, then the code exists but if rate R is larger than capacity C, the error probability is 1 and the length of the codeword becomes infinite.

3.2 Singleton Bound

It is preferable to have a large minimum distance d so that many errors can be corrected. Also, a large amount of codewords M would allow for efficient use of bandwidth when transmitting over a noisy channel. Unfortunately, increasing d tends to increase n or decrease M. The Singleton bound is an upper bound for M in terms of n and d. A code that satisfies the Singleton bound is called a MDS code (maximum distance separable). The Singleton bound can be written as

$$q^d \leq \frac{q^{n+1}}{M}$$

for the MDS code to obtain the largest possible value of d for a given n and M. Reed-Solomon codes are an important class of MDS codes (Trappe & Washington, 2006).

3.3 Maximum-likelihood Decoding

There are two principles of decoding. In hard-decision decoding the received bits are believed to be either 1 or 0 in binary transmission. The decoding is done bit by bit. In soft-decision decoding, the received codewords may contain samples of bits with many values, not just 1 or 0. Calculating the closest error-free codeword is more complicated but soft-decision decoding has better performance than the hard-decision decoding.

Assuming hard-decision decoding is used, the received codeword is decoded into its closest codeword measured by its smallest Hamming distance. This minimum probability of error principle is called Maximum-Likelihood or Minimum Distance Decoding (Geisel et al., 1990).

The model of the binary symmetric channel (BSC) (MacKay, 2003) in Figure 1.2 shows that the channel has binary input and output with an error probability, the channel is characterised by the following conditions if c is the transmitted code and r the received code:

$$\left.\begin{array}{ll} P(r=0|c=0) & = \text{1-p} \\ P(r=0|c=1) & = \text{p} \\ P(r=1|c=0) & = \text{p} \\ P(r=1|c=1) & = \text{1-p} \end{array}\right\} \text{ and } 0 \leq p \leq \frac{1}{2}$$

Comparing all received codewords r to all transmitted codewords c as a direct way of correcting errors would not be inefficient. This means storing all $2k$ code vectors and performing equally

as many comparisons for each received codeword, resulting in error vectors of which the vector with the smallest distance is probably the transmitted codeword. A more practical decoding method would be Syndrome decoding which will be described in Section 3.5.

3.4 Hamming Codes

Hamming constructed a code where he added three additional bits to four information bits. These additional or parity bits are chosen based on the information bits in the following manner:

$$p_1 = i_1 + i_2 + i_4$$
$$p_2 = i_1 + i_3 + i_4$$
$$p_3 = i_2 + i_3 + i_4$$

and form the codeword $c = (i_1, i_2, i_3, i_4, p_1, p_2, p_3)$. Hamming codes are block codes. This means that a fixed block of input data is processed into a fixed block of output data. A code is called a systematic code if the codeword starts with the information bits, followed by the parity bits, as shown in Figure 3.2. A non-systematic code has the information bits in a different order. The parity bits are the result of a modulo 2 addition, so if there is an even amount of bits, it gives 0 and 1 when there is an odd amount. If a single error occurs, i.e., a bit is flipped or reversed, the codeword no longer satisfies the equations.

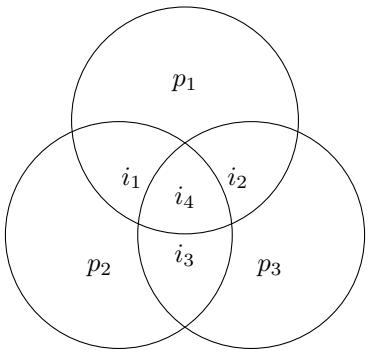

Figure 3.1: Relation between information and parity bits

Figure 3.2: An example of a systematic codeword of length n

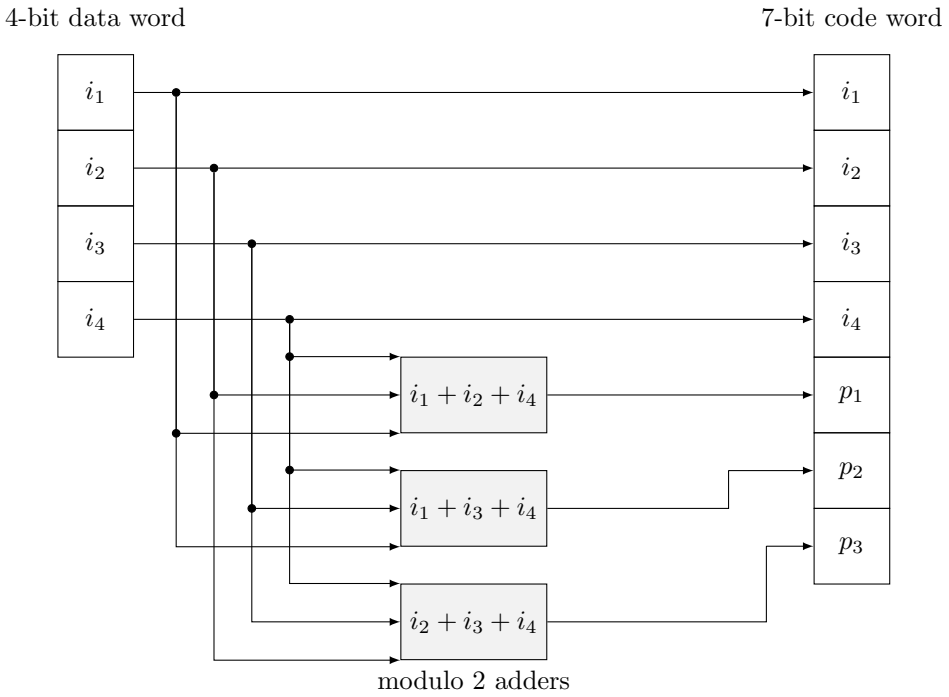

Figure 3.3: Hamming (7,4) encoder.

The decoder receives a seven-bit codeword $r = (i'_1, i'_2, i'_3, i'_4, p'_1, p'_2, p'_3)$. With an algebraic method known as syndrome decoding it is possible to determine the position of the error:

$$s_1 = p'_1 + i'_1 + i'_2 + i'_4$$
$$s_2 = p'_2 + i'_1 + i'_3 + i'_4$$
$$s_3 = p'_3 + i'_2 + i'_3 + i'_4$$

The three-bit syndrome (s_1, s_2, s_3) returns $(0, 0, 0)$ when a received codeword contains no errors. There are seven more possible syndromes, each corresponding to the position of the error in the received codeword. The decoder then inverts the detected bit to counter the error.

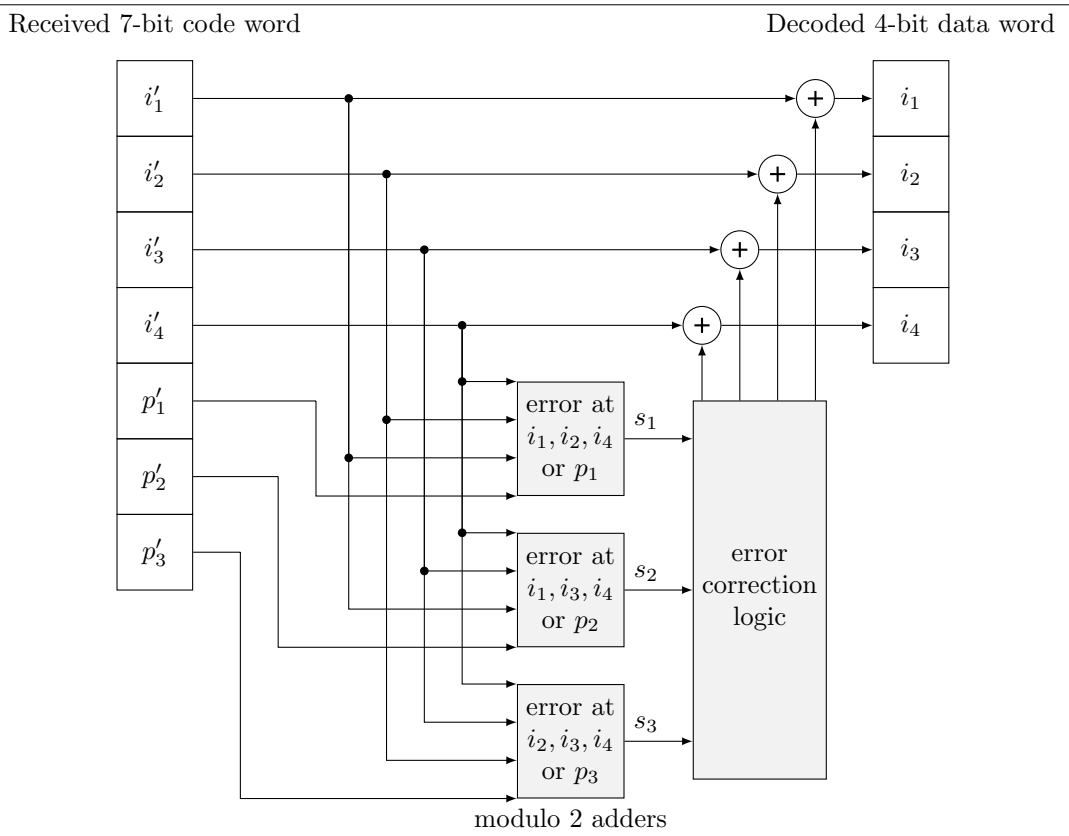

Figure 3.4: Hamming (7,4) decoder.

(Bose, 2008) considered the space of q-ary m-tuples, where every q-ary vector of length m can be represented by its endpoint in this space. Hence, we can represent every codeword as a point in this space, and all codewords at a Hamming distance of t or less would lie within the sphere centred at the codeword and with a radius of t.

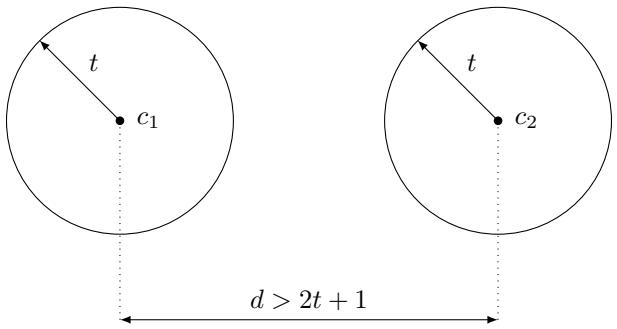

Figure 3.5: Decoding sphere

3.5 Syndrome Decoding

(Trappe & Washington, 2006) define linear (n, k) code of dimension k and length n over a field F as a k-dimensional subspace of F^n. For example, a linear binary code, of length n and dimension k is a set of 2^k binary codewords or n-tuples, such that the sum of any two codewords is always a codeword. To construct such a linear (n, k) code, we choose a $k \times n$ matrix known as generator matrix. The rows have to be linearly independent to produce unique codewords.

Generator matrix G is taken so that $G = [I_k, P]$, where I_k is the $k \times k$ identity matrix which determine the codewords and P is a $k \times (n - k)$ matrix that provides redundancy, the parity matrix. Now every codeword c of code C can be expressed as a linear combination of rows of G by $c = i \cdot G$. We can now calculate the generator matrix for a systematic representation. For example, a systematic Hamming (7,4) code has the following generator matrix:

$$G = [I_4|P] = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The parity check matrix is then calculated as

$$H = [-P^T|I_3] = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

For example, encoding information bits [1100] gives

$$\begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Decoding received codeword $c' = [1100011]$ with syndrome decoding results in [000] when no errors are detected. However, in our example an error was introduced in the fifth position of codeword $c' = [1100111]$, so we can expect a syndrome with non-zero elements.

$$c' \times H^T = \begin{bmatrix} 1 & 1 & 0 & 0 & \underline{1} & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ \underline{1} & \underline{0} & \underline{0} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

The value [100] can be looked up in parity check matrix H and tells that the error occurred in the fifth position from the left. Correction based on syndrome requires more steps and asks for a matrix of all single error vectors. Codeword $c = [1100011]$ and received codeword $c = [1100111]$ give an error vector of $e = [0000100]$ or $c = c' + e$. Since we already know that $s = c' \cdot H^T$ and an error-free $c \cdot H^T$ has a syndrome with all zero elements, we now substitute c' with $c + e$ because $c = c' + e$ is equivalent to $c' = c + e$ in binary.

$$
\begin{aligned}
s &= c' \cdot H^T \\
&= (c + e) \cdot H^T \\
&= c \cdot H^T + e \cdot H^T \\
&= 0 + e \cdot H^T \\
&= e \cdot H^T
\end{aligned}
$$

We can conclude that the syndrome solely depends on the error pattern and not on the transmitted codeword.

3.6 Cyclic Codes

Cyclic codes are widely used in data communication because their structure makes encoder and decoder circuitry simple. (Hill, 1986) defines code C as cyclic (n, k) code if C is a linear code of length n over a finite field and if any cyclic shift of a codeword is also a codeword. Thus,

$$(c_0, c_1, c_2, \ldots, c_{n-1}) \in C \text{ and } (c_{n-1}, c_0, c_1, \ldots, c_{n-2}) \in C.$$

Let $g(x)$ be the polynomial with the smallest degree. By dividing its highest coefficient, we may assume that the highest non-zero coefficient of $g(x)$ is 1. The polynomial $g(x)$ is called the generator polynomial for C, which must be a divisor of $x^n - 1$ (in a binary field this is equal to $x^n + 1$) with a degree of $n - k$. Subsequently, every cyclic code is a polynomial (Trappe & Washington, 2006). The encoder for cyclic codes is then

$$c(x) = i(x) \cdot g(x)$$

where $c(x)$ is the polynomial with degree $n-1$ of codeword $(c_0, c_1, c_2, \ldots, c_{n-1})$ which is calculated as

$$
\begin{aligned}
c(x) &= \sum_{i=0}^{n-1} c_i \cdot x^i \\
&= c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-1} x^{n-1}
\end{aligned}
$$

and $i(x)$ is the information polynomial of degree $k - 1$. Generator polynomial $g(x)$ must be of degree $n - k$.

3.7 BCH Codes

A BCH code is a cyclic polynomial code over a finite field with a particularly chosen generator polynomial. Hamming codes are the subset of BCH codes with $k = 2^m - 1 - m$ and have an error correction of 1. Generally, a family of t-error correcting codes defined over finite fields $GF(q)$, where $2t + 1 < q$, are BCH codes or RS codes (Hill, 1986). The main advantage of BCH codes is the ease with which they can be decoded using syndrome and many good decoding algorithms exist. A well-known decoding algorithm is the Berlekamp-Massey algorithm. This allows very simple electronic hardware to perform the task, making the need for a computer

unnecessary. This implies that a decoding device may be small and consume little power. BCH codes allow control over block length and acceptable error thresholds, which makes them very flexible. This indicates that code can be designed to meet custom requirements. Another reason they are important is that there exist good decoding algorithms that correct multiple errors. Hocquenghem, as well as Bose and Ray-Chaudhuri, discovered the class of BCH codes, but not the decoding. Peterson developed the first decoding algorithm in 1960 followed by refinement from Berlekamp, Massey and many others (Trappe & Washington, 2006).

3.7.1 Generating BCH Code

It is easy to generalise the construction of a t-error-correcting code of length $n = 2^m - 1$ over $GF(q) = \{0, 1, ..., q-1\}$ provided $2t + 1 \leq n \leq q - 1$. According to (Hill, 1986) it is not difficult to construct a binary BCH code over an extension field $GF(q^m)$. In order to obtain a cyclic code only the generator polynomial $g(x)$ is needed. For any integer $m \geq 3$ and $t < 2^{m-1}$, there exists a primitive BCH code with parameters:

$$\begin{aligned} n &= 2^m - 1 \\ n - k &\leq m \cdot t \\ d_{min} &\leq 2t + 1 \end{aligned}$$

Let α be a primitive n-th root of unity of $GF(2^m)$. For $1 \leq i \leq t$, let $m_{2i-1}(x)$ be the minimum polynomial of α_{2i-1}. The degree of $m_{2i-1}(x)$ is m or a factor of m. The generator polynomial $g(x)$ of a t-error-correcting primitive BCH codes of length $2^m - 1$ is given by

$$g(x) = LCM\{m_1(x), m_2(x), m_3(x), ..., m_{2t-1}(x), m_{2t}(x)\}$$

with LCM being Least Common Multiple, and because every even power of a primitive element has the same minimal polynomial as some odd power of the element, then $g(x)$ can be reduced to

$$g(x) = LCM\{m_1(x), m_3(x), ..., m_{2t-1}(x)\}$$

The degree of $g(x)$ is $m \cdot t$ or less and so is the number of parity check bits, therefore $n - k \leq m \cdot t$ (Lint, 1999).

Generally a code is a BCH code over $GF(q)$ with $m, n, d, c \in N$ chosen such that q is a prime power and $2 \leq d \leq n$. Also, m is the multiplicative order of q modulo n and n is not divisible by q, so the greatest common divisor of n and q is 1 (Lidl & Pilz, 1998). In special circumstances it is that,

- A BCH code with $c = 1$ is called a narrow-sense BCH code;
- A BCH code with $n = q^m - 1$ is called primitive;
- A narrow-sense BCH code with $n = q^m - 1$ is called a Reed-Solomon code.

The consecutive roots of the generator polynomial may run from $\alpha^c, \ldots, \alpha^{c+d-2}$ instead of $\alpha, \ldots, \alpha^{d-1}$. As before, let α be a primitive n-th root of unity in $GF(q^m)$, and let $m_i(x)$ be the minimal polynomial over $GF(q)$ of α_i for all i. The generator polynomial of the BCH code is defined as the least common multiple $g(x) = LCM\{m_c(x), \cdots, m_{c+d-2}(x)\}$ (Trappe & Washington, 2006).

3.7.2 Decoding BCH Code

BCH codes can be decoded in many way and it is most common that

- Syndromes values for are calculated for the received codeword;
- Error polynomials are calculated;
- Roots of these polynomials are calculated to obtain the location of errors;
- Error values are calculated at these locations.

Let code C be a binary BCH code with distance $d \geq 3$. C is a cyclic code of length n, with generating polynomial $g(x)$. There is a n-th root of unity α such that

$$g(\alpha^{k+1}) = g(\alpha^{k+2}) = 0$$

for some integer k.

$$\text{Let } H = \begin{bmatrix} 1 & \alpha^{(k+1)} & \alpha^{2(k+1)} & \dots & \alpha^{(n-1)(k+1)} \\ 1 & \alpha^{(k+2)} & \alpha^{2(k+2)} & \dots & \alpha^{(n-1)(k+2)} \end{bmatrix}$$

If $c = (c_0, \dots, c_{n-1})$ is a codeword, then polynomial $m(x) = c_0 + c_1 x + ... + c_{n-1} x^{n-1}$ is a multiple of $g(x)$, so

$$m(\alpha^{k+1}) = m(\alpha^{k+2}) = 0$$

This may be rewritten in terms of H:

$$c \cdot H^T = [c_0, \dots, c_{n-1}] \cdot \begin{bmatrix} 1 & 1 \\ \alpha^{(k+1)} & \alpha^{(k+2)} \\ \alpha^{2(k+1)} & \alpha^{2(k+2)} \\ \vdots & \vdots \\ \alpha^{(n-1)(k+1)} & \alpha^{(n-1)(k+2)} \end{bmatrix} = 0$$

H is not necessarily a parity matrix for C, however, it can correct an error.

Suppose codeword $c' = c + e$ is received with error vector $e = (e_0, ..., e_{n-1})$. Assuming that there is one error, the algorithm for correcting one error is to write $c' \cdot H^T = (s_1, s_2)$.

- If $s_1 = 0$ then there is either no error or more than one error and we stop here.
- If $s_1 \neq 0$, take $\frac{s_2}{s_1}$ which results in a power α^{j-1} of α.

The error is in position j and $e_j = 1$. Subtracting the error vector e from the received codeword c' gives the corrected codeword c'_r. For binary BCH codes it is only necessary to calculate the position, because the error value is always equal to 1. In non-binary BCH codes an additional error value polynomial is needed (Trappe & Washington, 2006).

3.8 Reed-Solomon codes

RS codes, which are BCH codes, are used in applications such as spacecraft communications, compact disc players, disk drives, and two-dimensional bar codes. According to (Bossert, 1999) the relationship between BCH and RS codes is such that RS codes comprise a subset of BCH codes and occasionally BCH codes comprise a subset of RS codes. (Lint, 1999) defines an RS code as a primitive BCH code of length $n = q - 1$ over $GF(q)$.

3.8.1 Generating a Reed-Solomon code

Let $GF(q)$ be a finite field with q elements and it generate a rather specific BCH code C over $GF(q)$ of length n, called a Reed-Solomon code. Let α be a primitive n-th root of unity of $GF(q)$ and let code C have a length of $n = q - 1$. Now take d so that $1 \leq d \leq n$ and the generator polynomial $g(x)$ is given by

$$\begin{aligned} g(x) &= \prod_{i=1}^{d-1}(x - \alpha^i) \\ &= (x - \alpha)(x - \alpha^2)\dots(x - \alpha^{d-1}) \end{aligned}$$

(Trappe & Washington, 2006) state that the minimum distance for C is at least d. Since $g(x)$ is a polynomial of degree $d - 1$, it has at most d non-zero coefficients. Therefore, the codeword corresponding to the coefficients of $g(x)$ has a weight of at most d. It follows that C has a weight of exactly d and the dimension of C is n minus the degree of $g(x)$

$$\begin{aligned} n - deg(g) &= n - (d - 1) \\ &= n + 1 - d \end{aligned}$$

Therefore, a Reed-Solomon code is a cyclic $(n, n+1-d, d)$ code with codewords corresponding to polynomials, where each $f(x)$ is a polynomial with coefficients in $GF(q)$ that cannot be factored into lower degree polynomials while assuming that the highest non-zero coefficient is 1:

$$g(x) \cdot f(x) \text{ with } deg(f) \leq n - d$$

It follows that there are q choices for each $n - d + 1$ coefficients of $f(x)$, and thus there are q $n-d+1$ codewords in code C. Therefore, an RS code is a MDS code since it makes the Singleton bound an equality.

Chapter 4

Visualisation

4.1 Bit Stream Encoding

4.2 Cross-interleaved Reed-Solomon Code

4.3 Decoding

Chapter 5

Summary and Conclusion

References

Blahut, R. (1983). *Theory and practice of error control codes*. Addison-Wesley Publishing Company.
https://books.google.fi/books?id=vuVQAAAAMAAJ

Bose, R. (2008). *Information theory, coding and cryptography*. Tata McGraw-Hill.
https://books.google.fi/books?id=AizEjrRIEHYC

Bossert, M. (1999). *Channel coding for telecommunications*. Wiley.
https://books.google.fi/books?id=MfdSAAAAMAAJ

Geisel, W., Aeronautics, U. S. N., Administration, S., & Center, L. B. J. S. (1990). *Tutorial on reed-solomon error correction coding*. National Aeronautics; Space Administration, Lyndon B. Johnson Space Center.
https://books.google.fi/books?id=Uys2AQAAMAAJ

Hill, R. (1986). *A first course in coding theory*. Clarendon Press.
https://books.google.fi/books?id=UTxjBX9lKoMC

Hocquenghem, A. (1959). Codes correcteurs d'erreurs. *Chiffers*, *2*, 147–156.

Lidl, R., & Pilz, G. (1998). *Applied abstract algebra*. Springer-Verlag.
https://books.google.fi/books?id=_49AmSWo4_AC

Lint, J. v. (1999). *Introduction to coding theory*. Springer-Verlag.
https://books.google.fi/books?id=tvQhRUFh7EwC

MacKay, D. (2003). *Information theory, inference and learning algorithms*. Cambridge University Press.
https://books.google.fi/books?id=AKuMj4PN_EMC

Massey, J. (1969). Shift-register synthesis and bch decoding. *IEEE Transactions on Information Theory*, *15*(1), 122–127. https://doi.org/10.1109/TIT.1969.1054260

Trappe, W., & Washington, L. (2006). *Introduction to cryptography: With coding theory*. Pearson Prentice Hall.
https://books.google.fi/books?id=PFDIQgAACAAJ

Wicker, S., & Bhargava, V. (1999). *Reed-solomon codes and their applications*. Wiley.
https://books.google.fi/books?id=3EK7EAAAQBAJ