



Fakultät Informatik

# Fehlertoleranz mit Reed-Solomon

Schriftlicher Bericht im Fach  
Aktuelle Entwicklungen im Computer Design

vorgelegt von

Jonas Lang

Matrikelnummer 363 0314

am 25. Juni 2024

© 2024

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

# Inhaltsverzeichnis

|  |           |
|--|-----------|
| <b>Abbildungsverzeichnis</b> . . . . .                           | <b>iv</b> |
| <b>Tabellenverzeichnis</b> . . . . .                             | <b>v</b>  |
| <b>1 Einleitung</b> . . . . .                                    | <b>1</b>  |
| 1.1 Motivation . . . . .   | 1         |
| 1.2 Zielsetzung der Arbeit . . . . .                             | 1         |
| <b>2 Entwicklung</b> . . . . .                                   | <b>3</b>  |
| <b>3 Theoretische Grundlagen</b> . . . . .                       | <b>5</b>  |
| 3.1 Polynom . . . . .  | 5         |
| 3.2 Lineare Gleichungssysteme . . . . .                          | 5         |
| 3.3 Endliche Körper . . . . .                                    | 6         |
| <b>4 Funktionsweise</b> . . . . .                                | <b>8</b>  |
| 4.1 Encodierung . . . . .  | 9         |
| 4.2 Decodierung . . . . .  | 9         |
| 4.3 Berlekamp-Welch-Algorithmus . . . . .                        | 11        |
| 4.4 Hardware-Implementierung . . . . .                           | 11        |
| <b>5 Anwendungen</b> . . . . .                                   | <b>13</b> |
| 5.1 Satelliten- und Weltraumkommunikation . . . . .              | 13        |
| 5.2 Broadcasting und digitale Fernsehtechnik . . . . .           | 13        |
| 5.3 Speichergeräte und optische Datenträger . . . . .            | 14        |
| 5.4 Datenübertragung und digitale Kommunikation . . . . .        | 14        |
| 5.5 Zweidimensionale Barcodes . . . . .                          | 14        |
| <b>6 Fazit und Ausblick</b> . . . . .                            | <b>16</b> |
| 6.1 Aktuelle Entwicklungen . . . . .                             | 16        |
| 6.2 Zukünftige Perspektiven . . . . .                            | 16        |
| 6.3 Fazit . . . . .  | 17        |
| <b>A Weiterführende Informationen</b> . . . . .                  | <b>18</b> |
| A.1 Reed-Solomon mit BCH-Schema . . . . .                        | 18        |
| A.2 Beweis des Hamming-Abstands von Reed-Solomon-Codes . . . . . | 18        |

## *Inhaltsverzeichnis*

|  |           |
|--|-----------|
| A.3 Vergleich mit anderen Codierungsverfahren . . . . .              | 19        |
| A.4 Beispielhafte Durchführung des Reed-Solomon-Verfahrens . . . . . | 20        |
| A.5 Umsetzungsparameter der verschiedenen Anwendungsfälle . . . . .  | 21        |
| A.6 Fehlertoleranz bei QR-Codes . . . . .                            | 22        |
| <b>Glossar . . . . .</b>   | <b>24</b> |
| <b>Literatur . . . . .</b>   | <b>25</b> |

# Abbildungsverzeichnis

|     |   |    |
|-----|---|----|
| 2.1 | Ausschnitt der Hierarchie von ECC . . . . .       | 4  |
| 4.1 | Ablauf der Kanalcodierung . . . . .               | 8  |
| 4.2 | Schaltung zur Polynomdivision . . . . .           | 12 |
| A.1 | Metadaten eines QR-Codes (Stufe L) . . . . .      | 23 |
| A.2 | Datensymbole in einem QR-Code (Stufe L) . . . . . | 23 |

# Tabellenverzeichnis

|     |   |    |
|-----|---|----|
| A.1 | Tabelle zum Vergleich verschiedener Codierungsverfahren . . . . . | 19 |
|-----|---|----|

# Kapitel 1

## Einleitung

### 1.1 Motivation

In der heutigen digitalen Welt ist die Zuverlässigkeit und Integrität von Daten von zentraler Bedeutung. Täglich werden riesige Mengen an Informationen über verschiedene Kommunikationskanäle übertragen und auf unterschiedlichsten Medien gespeichert. Dabei ist es unvermeidlich, dass einige Daten durch Rauschen, physische Beschädigungen oder andere Störfaktoren verfälscht werden. Die Gewährleistung der Genauigkeit und Verfügbarkeit von Informationen ist insbesondere in Bereichen der Telekommunikation, Datenarchivierung und digitalen Medien von entscheidender Bedeutung. Die Sicherstellung dieser Faktoren stellt jedoch eine ernsthafte Herausforderung an IT-Systeme dar. Fehlererkennungs- und -korrekturverfahren sind daher unverzichtbare Werkzeuge, um die Qualität und Zuverlässigkeit der übermittelten oder gespeicherten Daten sicherzustellen [1, Kapitel 1].

Um das zu erreichen, werden bei der Kanalcodierung die zu speichernden oder zu übertragenden Daten beim Encodierungsprozess mit Redundanz, also zusätzlichen Informationen, die zur Fehlererkennung und -korrektur dienen, angereichert. Im Decodierungsprozess wird an Hand der Redundanz überprüft, ob Fehler aufgetreten sind und ob diese korrigiert werden können. Dadurch kann die Integrität der empfangenen Daten bewertet werden [2, Kapitel 1].

Eine besonders effektive Methode zur Fehlerkorrektur sind die Reed-Solomon-Codes, also das ursprüngliche Verfahren und alle Weiterentwicklungen. Die ursprüngliche Version wurde 1960 von den Mathematikern Irving S. Reed und Gustave Solomon entwickelt [3]. Reed-Solomon-Codes zeichnen sich durch ihre Fähigkeit aus, so viele Fehler wie die Anzahl der hinzugefügten redundanten Informationen zu erkennen und sogar die Hälfte dieser zu korrigieren. In dieser Arbeit liegt der Fokus auf der Fehlerkorrektur.

### 1.2 Zielsetzung der Arbeit

Obwohl die Reed-Solomon-Codes in vielen alltäglichen Technologien wie CDs oder QR-Codes weit verbreitet sind, ist die zugrundeliegende Mathematik und konkrete Implementierung eher

unbekannt [4, Kapitel 1.5]. Ziel dieser Arbeit ist es, die Reed-Solomon-Codes vorzustellen und deren Funktionsweise zu erklären. Dazu wird nach einer kurzen historischen Betrachtung die der Kanalcodierung zu Grunde liegende Mathematik beschrieben. Anschließend wird die Funktionsweise des ursprünglichen Ansatzes der beiden Mathematiker und eine Weiterentwicklung, der Berlekamp-Welch-Algorithmus, erläutert. Des Weiteren wird beleuchtet, wie diese Verfahren in der Praxis Anwendung finden.

# Kapitel 2

## Entwicklung

Der ursprüngliche Reed-Solomon-Code wurde 1960 von den amerikanischen Mathematikern am Lincoln Laboratory des MIT entwickelt. Sie veröffentlichten das Paper „Polynomial codes over certain finite fields“, in dem das neu entworfene Fehlerkorrekturverfahren beschrieben wurde. Dabei handelt es sich um ein Vorwärtskorrektur-Verfahren (FEC), aus dem englischen „forward error correction“, welches eigenständig, ohne Rückfrage beim Absender, Fehler erkennen und beseitigen kann [2, Kapitel 1.1]. Im Gegenzug dazu gibt es die Verfahren, die Fehler nur erkennen können und daraufhin fehlerhafte Daten erneut anfragen. Deshalb werden sie „automatic repeat request“ (ARQ)-Verfahren genannt [5, Kapitel 2.1].

Durch diese Veröffentlichung wurde eine neue Klasse von Fehlerkorrektur-Codes (ECC) geschaffen. Reed-Solomon-Codes gehören zu der Gruppe der linearen, zyklischen Block-Codes. Lineare Blockcodes teilen zu codierende Daten in Blöcke mit fester Länge  $n$  und verarbeiten diese in einer endlichen algebraischen Struktur [2, Kapitel 3]. Bei zyklischen Codes ist diese eine endliche Algebra (siehe Abschnitt 3.3 auf Seite 6). Ein Ausschnitt der Code-Hierarchie ist in Abbildung 2.1 dargestellt.

Allerdings war fast 25 Jahre lang für das effektive Reed-Solomon-Verfahren kein effizienter Decodieralgorithmus bekannt. In der Zwischenzeit wurde aber weiter geforscht und -entwickelt.

1963 stellte J. J. Stone einen Reed-Solomon-Code vor, der auf dem Schema des Bose-Chaudhuri-Hocquenghem (BCH)-Verfahrens von 1959 basiert [31]. Trotz des Namens handelt es sich dabei aber um ein eigenständiges Verfahren, welches separat vom ursprünglichen Ansatz weiterentwickelt wurde (siehe Anhang A.1 auf Seite 18). Nachdem für die BCH-Codes 1969 ein effizienter Algorithmus, der so genannte Berlekamp-Massey-Algorithmus, gefunden wurde, war damit auch der Einsatz des auf dem BCH-Verfahren basierenden Reed-Solomon-Codes in der Praxis möglich [6], [7].

Anwendung fand dieses Verfahrens zum ersten Mal im Jahr 1977 und zwar beim Voyager Programm der NASA [4, Kapitel 3.3]. Dadurch konnte eine robustere Kommunikation zwischen dem Raumfahrzeug und dem Raumfahrtkontrollzentrum bei einer großen Entfernung zur Erde



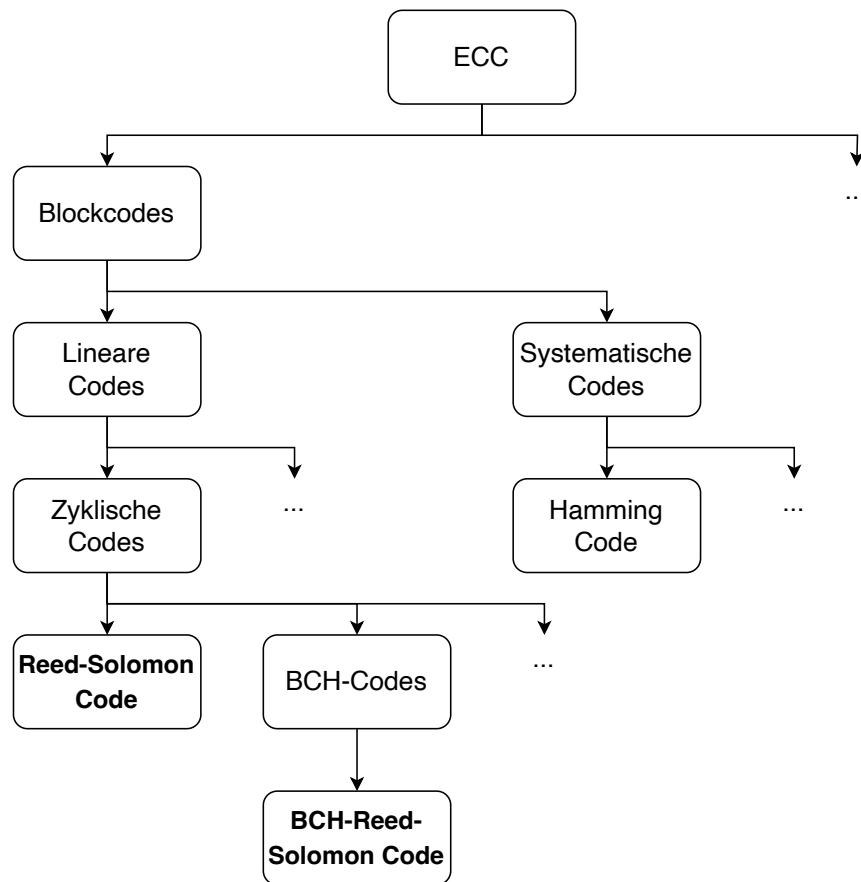


Abbildung 2.1: Ausschnitt der Hierarchie von ECC

gewährleistet werden [8, Kapitel 6.2]. Das erste kommerzielle, weit verbreitete und an den Endkunden gerichtete Produkt, in dem das Reed-Solomon-Verfahren eingesetzt wurde, war 1982 die Compact Disc (CD) [9].

Im Jahr 1986 ermöglichte der Berlekamp-Welch-Algorithmus die entscheidende Weiterentwicklung [10]. Entwickelt von Elwyn Berlekamp und Lloyd Welch verbesserte dieser Algorithmus die Effizienz des ursprünglichen Reed-Solomon-Schemas erheblich und förderte somit die Verbreitung der Reed-Solomon-Codes in verschiedenen Anwendungsbereichen. Eine detaillierte Beschreibung dieser findet sich in Kapitel 5 auf Seite 13.

Die kontinuierliche Weiterentwicklung und Anwendung der Reed-Solomon-Codes in verschiedenen Bereichen zeigt die Bedeutung und Vielseitigkeit dieses Fehlerkorrekturverfahrens. Um die Funktionsweise und die mathematischen Prinzipien dieses Codes in seiner Tiefe zu erfassen, ist es unabdingbar, sich mit den theoretischen Grundlagen auseinanderzusetzen. Im nächsten Kapitel werden daher die mathematischen Konzepte und Algorithmen erläutert, die dem Reed-Solomon-Code zugrunde liegen.

# Kapitel 3

## Theoretische Grundlagen

### 3.1 Polynom

Ein Polynom ist ein Ausdruck der Form

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0,$$

wobei die Koeffizienten  $a_0, a_1, \dots, a_n$  reelle oder komplexe Zahlen und  $x$  eine Variable sind. Der höchste Exponent  $n$ , bei dem der Koeffizient  $a_n \neq 0$  ist, wird als der Grad des Polynoms  $\deg(p)$  bezeichnet.

Es gibt neben dieser Standardform verschiedene Möglichkeiten, Polynome darzustellen. In dieser Arbeit wird auch die faktorisierte Form eines Polynoms benötigt.

Ein Polynom kann oft in ein Produkt von Linearfaktoren zerlegt werden, wenn seine Wurzeln (auch Nullstellen genannt) bekannt sind. Wenn  $r_1, r_2, \dots, r_n$  die Wurzeln des Polynoms sind, dann kann es wie folgt geschrieben werden:

$$p(x) = a_n (x - r_1)(x - r_2) \cdots (x - r_n),$$

wobei  $a_n$  der führende Koeffizient ist. Diese Form wird häufig auch als Nullstellenform bezeichnet [11, Kapitel 49].

Durch die verschiedenen Darstellungsformen können Polynome je nach Anwendung und Problemstellung unterschiedlich analysiert und interpretiert werden.

### 3.2 Lineare Gleichungssysteme

Ein lineares Gleichungssystem besteht aus mehreren linearen Gleichungen, die gleichzeitig erfüllt sein müssen. Ein solches System kann eine oder mehrere Unbekannte haben und wird allgemein

durch mehrere algebraische Ausdrücke dargestellt. Eine lineare Gleichung in  $n$  Variablen hat die Form:

$$a_0 + a_1x + a_2x_2 + \cdots + a_nx_n = b,$$

wobei  $a_1, a_2, \dots, a_n$  die Koeffizienten,  $x_1, x_2, \dots, x_n$  die Variablen und  $b$  eine Konstante sind. Die Variablen  $x_1, x_2, \dots, x_n$  haben alle die Potenz 1.

Zum Lösen dieser Gleichungssysteme gibt es verschiedene Lösungsverfahren, die hier nicht näher erläutert werden. Die Lösungen von Gleichungssystemen können wie folgt klassifiziert werden:

1. Eindeutige Lösung: Es gibt genau eine Lösung.
2. Mehrere Lösungen: Es gibt mehr als eine Lösung, oft dargestellt als Elemente einer Lösungsmenge  $L$ .
3. Keine Lösung: Es gibt keine Kombination von Werten, die alle Gleichungen gleichzeitig erfüllt.

Ein lineares Gleichungssystem ist genau dann eindeutig lösbar, wenn die Anzahl der Unbekannten gleich der Anzahl der Gleichungen ist.

Gleichungssysteme sind grundlegende Werkzeuge in der Mathematik und angewandten Wissenschaften, die eine Vielzahl von Problemen modellieren und lösen können.

## 3.3 Endliche Körper

Endliche Körper spielen eine wichtige Rolle in der Kryptographie und der Codierungstheorie. Ein endlicher Körper oder Galois-Körper ist eine Menge mit einer endlichen Anzahl von Elementen, auf der die Grundoperationen Addition, Subtraktion, Multiplikation und Division definiert sind.

Für jede Primzahl  $p$  ist der Restklassenring  $\mathbb{Z}/p\mathbb{Z}$  ein Körper und wird mit  $\mathbb{Z}_p$  oder  $GF(p)$  (vom englischen Galois field) bezeichnet. Jeder endliche Körper enthält  $\mathbb{Z}_p$  als einen Unterkörper. Er ist damit insbesondere ein Vektorraum über  $\mathbb{Z}_p$  und als solcher isomorph zu  $\mathbb{Z}_p^r$  für  $r \in \mathbb{N}$ . Er hat genau  $p^r$  Elemente.

Man kann zeigen, dass es bis auf Isomorphie genau einen Körper mit  $q = p^r$  Elementen gibt, dieser heißt  $GF(q) = GF(p^r)$ . Er ist eine einfache Erweiterung von  $\mathbb{Z}_p$ , d. h., es gibt ein irreduzibles Polynom  $f \in \mathbb{Z}_p[x]$  vom Grad  $r$ , so dass  $GF(p^r) \cong \mathbb{Z}_p[x]/(f)$ .

Im endlichen Körper  $GF(2^r)$  beispielsweise, auf dem die Reed-Solomon-Codes basieren, sind die Elemente Polynome vom Grad  $r - 1$  mit Koeffizienten aus  $\mathbb{Z}_2 = \{0, 1\}$ . Die Addition erfolgt koeffizientenweise modulo 2, während die Multiplikation durch die Multiplikation der Polynome mit anschließendem Reduzieren modulo  $p(x)$  durchgeführt wird [12, Kapitel 1.1].

Diese Eigenschaften sind wesentlich für die Implementierung der Fehlerkorrekturmechanismen der Reed-Solomon-Codes, da die Berechnungen auf endlichen Systemen durchgeführt werden. Die Struktur dieser Felder ermöglicht die effiziente Durchführung der mathematischen Operationen, die zur Erkennung und Korrektur von Fehlern in den Daten erforderlich sind.

# Kapitel 4

## Funktionsweise

Obwohl Daten im Computer als Bits vorliegen, werden beim Reed-Solomon-Verfahren die zu übermittelnden oder zu speichernden Daten nicht direkt als Bitstrom verarbeitet. Die zu codierenden Nachrichten werden in Symbole  $m_i$  der Länge 8 Bit zusammengefasst. Die Anzahl dieser Symbole ist dann die Nachrichtenlänge  $k$  [13]. Eine Nachricht  $m$  hat also die Form  $[m_0 m_1 m_2 \dots m_{k-1}]$ .

Zum Beispiel wird eine Nachricht mit dem Bitmuster 0010100110100101 in 00101001 10100101 zerlegt und einzeln in Symbole übersetzt (hier Dezimaldarstellung) 41 165. Hier wäre die Nachrichtenlänge 2.

Nach Hinzufügen der Redundanzsymbole durch Encodieren entsteht ein Block der Länge  $n$ . Die Spezifikation der Ausprägung des eingesetzten Reed-Solomon-Codes wird typischerweise als  $RS(n, k)$  dargestellt. Beim Empfänger kommt dann nach dem Decodieren die eventuell fehlerbereinigte Originalnachricht an [14]. Dieser Ablauf ist in der Abbildung 4.1 dargestellt.

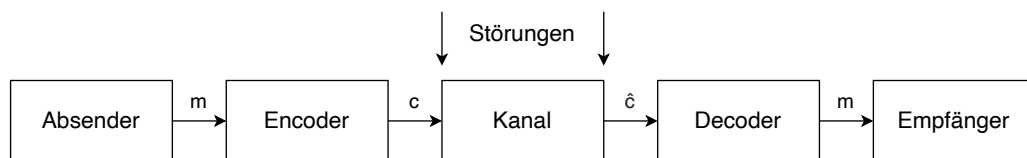


Abbildung 4.1: Ablauf der Kanalcodierung

Die Berechnungen erfolgen dabei in dem endlichen Körper  $GF(2^r)$  über  $\mathbb{Z}_2$ , dem Lieblingsring im Computer, der genau  $2^r = n$  Elemente hat. Diese Elemente lassen sich als Potenzen einer primitiven Einheitswurzel über einem geeigneten irreduziblen Polynom generieren [11, Kapitel 3]. Diese mathematischen Grundlagen gelten für alle Reed-Solomon Varianten.

Eine Erklärung des auf dem BCH-Schema basierenden Reed-Solomon-Codes findet sich in Anhang A.1 auf Seite 18. Die folgenden Ausführungen zur Encodierung (Abschnitt 4.1) und Decodierung (Abschnitt 4.2) beziehen sich auf den ursprünglichen Ansatz von 1960.

## 4.1 Encodierung

Aus einer Nachricht  $m = [m_0 m_1 m_2 \dots m_{k-1}]$  entsteht das Polynom

$$p(x) = \sum_{i=0}^{k-1} m_i x^i = m_0 + m_1 x + m_2 x^2 + \dots + m_{k-1} x^{k-1}$$

mit den Nachrichtensymbolen  $m_i$  als Koeffizienten der einzelnen Summanden. Dieses Polynom wird als Nachrichtenpolynom bezeichnet [3]. Dieses Nachrichtenpolynom wird bei anderen Reed-Solomon-Varianten mit verschiedenen Methoden, zum Beispiel der Lagrange-Interpolation, generiert [10].

Für alle Nachrichten wird ein Codewort  $c = [c_0 c_1 c_2 \dots c_{n-1}]$  der Länge  $n = k + 2t$  generiert, wobei  $k$  die Länge der ursprünglichen Nachricht und  $2t$  die Anzahl der redundanten Symbole ist [15]. Dadurch können  $2t$  Fehler erkannt und  $t = \lfloor \frac{n-k}{2} \rfloor$  Fehler korrigiert werden (Beweis siehe Anhang A.2 auf Seite 18). Das Codewort entsteht durch das Einsetzen von  $n$  festen Werten in das Nachrichtenpolynom.

$$c_0 = p(0); c_i = p(\alpha^i) \quad (i \in \{1; \dots; n-2\})$$

Diese Werte  $[0 \alpha^0 \alpha^1 \alpha^2 \dots \alpha^{n-2}]$  sind die Potenzen einer primitiven Einheitswurzel des endlichen Körpers  $GF(2^r) = GF(n)$  zuzüglich der 0, also alle Elemente des Körpers [11, Kapitel 3]. Dieses Codewort wird dann gespeichert oder übertragen. Allerdings können im Kanal vor dem Decodieren Störungen das Codewort verfälschen.

## 4.2 Decodierung

Aus dem Codewort  $c = [c_0 c_1 c_2 \dots c_{n-1}]$  wird durch Decodieren die ursprüngliche Nachricht wiederhergestellt. Dazu muss die Encodierung rückgängig gemacht werden, indem für jedes Symbol die Encodierungsgleichung aufgestellt wird. Allerdings ist nun  $p(x)$  unbekannt und soll mit Hilfe der  $\alpha^i$  und der empfangenen  $c_i$  durch  $c_0 = p(0)$  und  $c_i = p(\alpha^{i-1})$  berechnet werden [3]. Dadurch entsteht ein Gleichungssystem mit  $n$  Gleichungen und  $k$  Unbekannten.

$$\begin{aligned} c_0 &= p(0) &= m_0 \\ c_1 &= p(\alpha^0) &= m_0 + m_1 &+ m_2 &+ \dots + m_{k-1} \\ c_2 &= p(\alpha^1) &= m_0 + m_1 \alpha &+ m_2 (\alpha)^2 &+ \dots + m_{k-1} (\alpha)^{k-1} \\ c_3 &= p(\alpha^2) &= m_0 + m_1 \alpha^2 &+ m_2 (\alpha^2)^2 &+ \dots + m_{k-1} (\alpha^2)^{k-1} \\ &&&\dots \\ c_{n-1} &= p(\alpha^{n-2}) &= m_0 + m_1 \alpha^{n-2} &+ m_2 (\alpha^{n-2})^2 &+ \dots + m_{k-1} (\alpha^{n-2})^{k-1} \end{aligned}$$

Die unbekannten Koeffizienten des Polynoms sind die gesuchten Nachrichtensymbole  $m_i$ . Da  $n > k$ , reichen  $k$  dieser Gleichungen aus, um das Gleichungssystem eindeutig zu lösen und so die Nachricht zu erhalten.

Allerdings ist das Codewort durch Störungen im Kanal möglicherweise fehlerbehaftet. Einige Symbole von  $c$  können also verändert worden sein, sodass ein  $\hat{c}$  entsteht und die dazugehörige Gleichung falsch ist [15]. Werden eine oder mehrere falsche Gleichungen in einem Gleichungssystem verwendet, liefert dieses veränderte Koeffizienten und somit eine falsche Lösung  $\hat{m}$ .

Wie viele Fehler und an welchen Stellen diese aufgetreten sind, lässt sich a priori nicht sagen. So reicht es also nicht aus, nur  $k$  Gleichungen zum Lösen des Gleichungssystems zu verwenden, da in diesen  $k$  Gleichungen bis zu  $2t$  Fehler enthalten sein können. Um eine korrekte Lösung des Gleichungssystems zu finden, müssen alle  $\binom{n}{k}$  möglichen Gleichungssysteme gelöst werden. So ergeben sich mindestens  $\binom{n-t}{k}$  Gleichungssysteme mit identischer Lösung [3]. Die Lösung der restlichen, fehlerbehafteten Gleichungssysteme variiert und gleiche Ergebnisse treten selten auf. Es wird gezählt, wie viele Gleichungssysteme eine bestimmte Lösung liefern und so wird das Ergebnis, welches am häufigsten aufgetreten ist, ausgewählt. So kann die Nachricht  $m$  aus dieser Lösung für  $[m_0 m_1 m_2 \dots m_{k-1}]$  wiederhergestellt werden. Bei dem Verfahren wird vorausgesetzt, dass nie mehr als  $2t$  Fehler auftreten, da sonst das eben beschriebene Lösungs-Auswahlverfahren fehlschlagen würde. Eine beispielhafte Durchführung dieses Verfahrens findet sich in Anhang A.4 auf Seite 20.

Es gibt kein Verfahren, das mehr Fehler korrigieren kann als der Reed-Solomon-Code. Das beweist der Hamming-Abstand, eine Kenngröße für die Effektivität von Codierungsverfahren, der bei Reed-Solomon  $n - k + 1$  ist [3]. Unter dem Hamming-Abstand eines Codes versteht man das Minimum aller Abstände zwischen verschiedenen Wörtern innerhalb des Codes. Bei Codes mit Hamming-Abstand  $h$  können alle  $(h - 1)$ -Bit-Fehler erkannt werden, also bei Reed-Solomon  $n - k = 2t$ . Ein ausführlicher Beweis dazu findet sich in Anhang A.2 auf Seite 18. Das Verfahren schafft demnach den optimalen Kompromiss zwischen der Länge der hinzugefügten Redundanz und der Fähigkeit, Fehler zu erkennen und zu korrigieren. Die Singleton-Schranke ist die Obergrenze für den Hamming-Abstand und beträgt  $n - k + 1$  [2, Kapitel 3.3]. Also kann kein Verfahren effektiver agieren als die Reed-Solomon-Codes.

Im Anhang A.3 auf Seite 19 findet sich eine Vergleichstabelle, in der der Reed-Solomon-Code mit dem Hamming-Code und dem BCH-Code hinsichtlich ihrer Effektivität anhand von Werten verglichen wird. Der Hamming-Code ist zwar ineffektiver, aber bei Einzelbitfehlern benutzerfreundlicher [16]. Der BCH-Code wird, obwohl er im Verhältnis von Redundanz zu Fehlern nicht so effektiv ist, bei einigen Anwendungen eingesetzt, da er kostengünstiger zu implementieren ist [12, Kapitel 3].

In den 1960ern war dieses effektive Verfahren leider noch nicht effizient einsetzbar, da das Lösen der Gleichungssysteme für größere Werte für  $n$  und  $k$  lange dauert [15]. Wenn man beispielsweise

eine typische Konfiguration  $RS(255, 223)$  aus dem Voyager-Programm annimmt, folgen daraus  $\binom{255}{223} \approx 5 \cdot 10^{40}$  Gleichungssysteme, die gelöst werden müssen. Bei einer Lösungsdauer von einer Millisekunde pro Gleichungssystem, wären das immer noch eine Gesamtdauer von ca.  $1,61 \cdot 10^{30}$  Jahren. So kann man leicht nachvollziehen, warum dieses Verfahren in dieser Form nicht effizient lösbar ist.

### 4.3 Berlekamp-Welch-Algorithmus

Der Berlekamp-Welch-Algorithmus bietet eine Lösung dieses Problems. Dazu wird ein Fehlerlokalisierungspolynom  $e(x) = (x - e_1)(x - e_2) \cdots (x - e_t)$  mit  $\deg(e) = t$  definiert, wobei die  $e_i$  die Stellen sind, an denen Fehler aufgetreten sind [11]. So ist  $e(x)$  an den Fehlerstellen gleich 0. Außerdem wird  $q(x) = p(x)e(x)$  mit  $\deg(q) = \deg(p) + \deg(e) = k - 1 + t$  gebildet. Umgeformt sind die zu bestimmenden Polynome

$$\begin{aligned} e(x) &= 1x^t + b_{t-1}x^{t-1} + \cdots + b_1x + b_0 \\ q(x) &= a_{k+t-1}x^{k+t-1} + a_{k+t-2}x^{k+t-2} + \cdots + a_1x + a_0, \end{aligned}$$

welche zusammen  $k + 2t$  unbekannte Koeffizienten haben. Die  $k + 2t$  Symbole des Codeworts liefern also genau genug Informationen um diese Koeffizienten eindeutig zu bestimmen, indem man  $k + 2t$  Gleichungen der Form  $q(i) = e(i)$  aufstellt, wobei  $1 \leq i \leq k + 2t$ . Mit diesen gegebenen Koeffizienten sind  $q(x)$  und  $e(x)$  bekannt, und es kann mit Polynomdivision  $\frac{p(x)}{e(x)}$  wieder das ursprüngliche Polynom  $p(x)$  berechnet werden. Damit ist die Nachricht mit den Koeffizienten, die sich durch die Polynome ergeben, decodiert [17].

### 4.4 Hardware-Implementierung

Um dieses Fehlerkorrekturverfahren in möglichst vielen Szenarien anwenden zu können, ist es wichtig, dass die Verwendung die Anwendungsfälle kaum beeinträchtigt. Deshalb wird häufig auf eine spezielle Hardware-Implementierung gesetzt, um den Einsatz auch auf kleinen Geräten wie Mikrokontrollern zu ermöglichen. Da viele Reed-Solomon-Codes auf der Polynomdivision basieren, gibt es dafür dedizierte Hardwarekomponenten [15]. Eine solche Implementierung zur Polynomdivision, wie in Abbildung 4.2 auf Seite 12 abgebildet, wird im Folgenden näher beschrieben. Die Schaltung arbeitet mit Galois-Addierern  $ga_i$  und Galois-Multiplizierern  $gm_i$  und ist rückgekoppelt, das heißt, die Berechnungen basieren auf dem zuvor durchgeführten Berechnungsschritt [7]. Am Eingang  $p(x)$  werden jeweils die Koeffizienten des zu dividierenden Polynoms angelegt, am Ausgang  $q(x)$  kommen die Koeffizienten des Ergebnisses an, jeweils die Koeffizienten mit dem höchsten Exponenten zuerst. Die  $gm_i$  enthalten jeweils den  $i$ -ten Koeffizienten des Divisors und multiplizieren diesen mit ihrem Eingang. Bei  $gm_0$  handelt es sich um den Koeffizienten



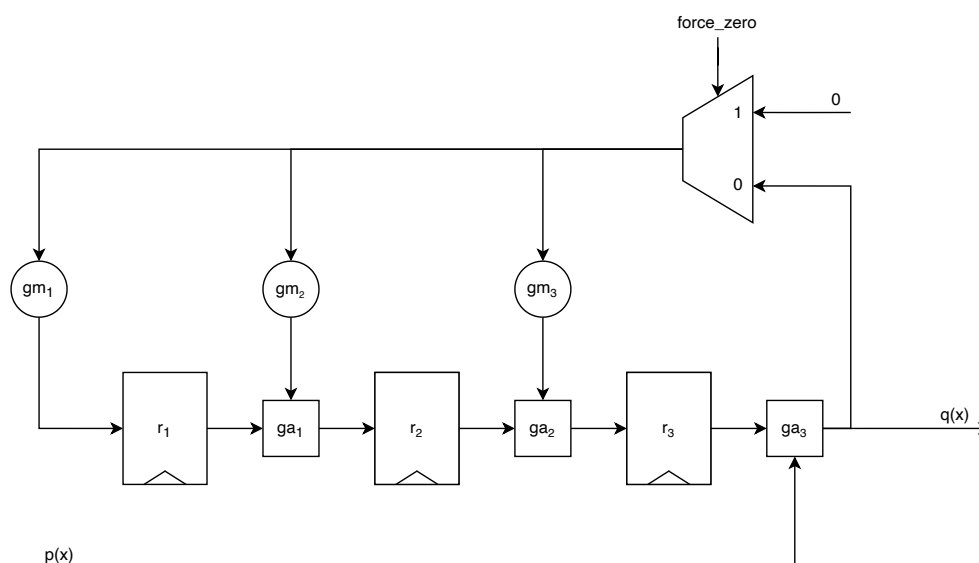


Abbildung 4.2: Schaltung zur Polynomdivision

von  $x^0$ . Die Addierer addieren ihre jeweiligen Eingänge und geben sie an ihren Ausgang weiter. Dabei werden die Reste eines Divisionsschrittes in den Registern  $r_i$  gespeichert und sequenziell in den folgenden Iterationen weiterverwendet, so wie man die Polynomdivision auch mit Stift und Papier durchführen würde. Nachdem am Eingang  $p(x)$  alle Koeffizienten angelegt wurden, wird **force\_zero** auf 1 gesetzt. So können die sich noch in den Registern befindlichen Reste ohne weitere Berechnungen sequentiell ausgegeben werden. Die Schaltung in der Abbildung ist der Übersichtlichkeit halber nur für einen Divisor mit einem Polynom aus drei Summanden dargestellt. In der Realität müssten, auf das konkrete Verfahren angepasst, entsprechend mehr Komponenten verbaut sein.

Durch das Verwenden von Galois-Multiplizierern und -Addierern entsteht ein Vorteil gegenüber der klassischen Arithmetik, denn die Ergebnisse bleiben innerhalb dieses Körpers und benötigen also immer die gleiche Anzahl an Speicherplatz [11, Kapitel 3]. Dadurch können die Komponenten wie Register und verbindende Busleitungen auf eine konkrete Bit-Anzahl angepasst werden mit dem Effekt, dass keine kostspielige Hardware verbaut sein muss und die Komponenten optimal ausgenutzt wird. Außerdem können diese Schaltungen mit einfachen logischen Gattern umgesetzt werden [18], [19]. Ein Galois-Addierer wird zum Beispiel durch ein XOR-Gatter implementiert.

Mit Hilfe dieser oder ähnlicher Schaltungen und dem in Abschnitt 4.3 auf Seite 11 genannten Algorithmus war es möglich, Reed-Solomon-Codes in verschiedenen Einsatzbereichen zu verwenden, wie im nächsten Kapitel erläutert wird.

# Kapitel 5

## Anwendungen

Reed-Solomon-Codes haben sich als äußerst vielseitig und effektiv in einer Vielzahl von Anwendungen erwiesen, insbesondere in Bereichen, die eine hohe Zuverlässigkeit und Robustheit bei der Datenübertragung und -speicherung erfordern [20]. Im Folgenden werden einige der wichtigsten Anwendungsgebiete dieses Fehlerkorrekturverfahren beschrieben. Die konkreten Umsetzungsparameter finden sich im Anhang A.5 auf Seite 21, sofern sie verfügbar sind.

### 5.1 Satelliten- und Weltraumkommunikation

Eine der frühesten Anwendungen der Reed-Solomon-Codes war im Bereich der Satelliten- und Weltraumkommunikation. Ein bemerkenswertes Beispiel ist das Voyager-Programm der NASA. Seit 1977 werden Reed-Solomon-Codes verwendet, um die Kommunikation zwischen den Voyager-Raumfahrzeugen und der Erde zu sichern. Andere Projekte mit Reed-Solomon-Codes im Einsatz sind zum Beispiel der *Mars Pathfinder* und die Raumsonde *Galileo* [4, Kapitel 3].

Die enorme Entfernung der Raumfahrzeuge zur Erde stellt eine besondere Herausforderung an die Datenintegrität. Durch kosmische Strahlung und andere Störeinflüsse werden Übertragungsfehlern verursacht, die Reed-Solomon-Codes korrigieren können und so zur erfolgreichen Übermittlung von wissenschaftlichen Daten über große Distanzen beitragen [8, Kapitel 5].

### 5.2 Broadcasting und digitale Fernsehtechnik

Im Bereich des Broadcasting, insbesondere bei der digitalen Fernsehtechnik, spielen Reed-Solomon-Codes eine entscheidende Rolle. Sie werden verwendet, um die Qualität und Zuverlässigkeit von digitalen TV-Signalen zu verbessern. Durch die Implementierung von Reed-Solomon-Codes können Übertragungsfehler, die durch atmosphärische Störungen oder andere Übertragungsprobleme entstehen, effektiv korrigiert werden, was zu einer stabileren und hochwertigeren Signalübertragung führt. Genutzt wird das Reed-Solomon-Verfahren beispielsweise von dem amerikanischen Standard Advanced Television Systems Committee (ATSC) und dem europäischen

Pendant Digital Video Broadcasting (DVB) [21]. DVB beinhaltet verschiedene Standards, welche mittlerweile andere Verfahren, wie zum Beispiel BCH-Codes verwenden. Beispielhaft für ein auf Reed-Solomon basierendes Protokoll ist DVB-H [22].

### 5.3 Speichergeräte und optische Datenträger

Reed-Solomon-Codes sind bei der Sicherstellung der Datenintegrität beim Speichern von Daten essentiell. Bei optischen Datenträgern, die seit der Einführung der Compact Disc (CD) im Jahr 1982 weit verbreitet sind, werden Reed-Solomon-Codes zur Fehlerkorrektur bei der Speicherung und Wiedergabe von digitalen Audio- und Videodaten verwendet. Dazu zählen neben den CD auch die Digital Video Disc (DVD) und die Blu-Ray-Disc. Diese Codes können Fehler erkennen und korrigieren, die durch Kratzer, Staub oder andere physische Beschädigungen an den Discs verursacht werden [9].

In RAID-Systemen (Redundant Array of Independent Disks), zum Beispiel verwendet für die Aufbewahrung von Backups, ermöglichen sie die Korrektur von Fehlern, wodurch ein Ausfall eines physischen Laufwerks nicht zum Verlust der auf dem System gespeicherten Daten führt. Diese RAID-Systeme bieten verschiedene Modi zur redundanten Datenspeicherung. Eine davon ist RAID6, welches auf Reed-Solomon-Codes basiert [23].

### 5.4 Datenübertragung und digitale Kommunikation

Reed-Solomon-Codes finden auch breite Anwendung in der digitalen Kommunikation, einschließlich der Datenübertragung über das Internet und in Mobilfunknetzen. Sie werden eingesetzt, um die Integrität von Datenpaketen zu gewährleisten, die über potenziell fehleranfällige Kanäle übertragen werden. Reed-Solomon-Codes verwendende Standards sind z.B. WiMAX und DSL. Allerdings ist die genaue Umsetzung dieser Protokolle nicht öffentlich [24, Kapitel 2].

### 5.5 Zweidimensionale Barcodes

Reed-Solomon-Codes sind auch in der Optoelektronik weit verbreitet. Beispiele hierfür sind MaxiCode, Datamatrix, AztecCode und QR-Code. Diese Codes nutzen die Fehlerkorrekturfähigkeiten von Reed-Solomon, um sicherzustellen, dass die gespeicherten Informationen auch dann korrekt ausgelesen werden können, wenn Teile des Codes beschädigt oder verdeckt sind. Dies ist besonders wichtig in Anwendungen, bei denen die Zuverlässigkeit der Datenlesung entscheidend ist, wie z.B. in der Logistik, im Einzelhandel und in Fragen der Sicherheit [25, Kapitel 3]. Bei QR-Codes gibt es verschiedenen Varianten von „Low“ bis „High“, welche unterschiedlich viel

Fehlertoleranz besitzen [26]. Wie QR-Codes diese Fehlertoleranz umsetzen, ist ausführlich in Anhang A.6 auf Seite 22 beschrieben.

# Kapitel 6

## Fazit und Ausblick

### 6.1 Aktuelle Entwicklungen

Reed-Solomon-Codes bleiben auch heute eine relevante Technologie in der Fehlerkorrektur und -detektion, obwohl sie in den letzten Jahrzehnten aus Kostengründen durch neue Methoden ergänzt und teilweise ersetzt wurden [21].

Allerdings wird die Integration von Reed-Solomon-Codes in modernen Kommunikations- und Speichertechnologien immer bedeutender. Insbesondere im Bereich der drahtlosen Kommunikation und der Netzwerkcodierung werden Reed-Solomon-Codes in Kombination mit anderen Fehlerkorrekturmethode eingesetzt, um eine höhere Zuverlässigkeit zu gewährleisten [27, Kapitel 1].

### 6.2 Zukünftige Perspektiven

In der Zukunft werden Reed-Solomon-Codes weiterhin eine wichtige Rolle in der Datenübertragung und -speicherung spielen, insbesondere in Kombination mit anderen Technologien. Die Entwicklungen im Bereich der Quantenkommunikation und Quantencomputing bieten neue Möglichkeiten, in denen klassische Fehlerkorrekturverfahren wie Reed-Solomon-Codes integriert werden könnten, um Systeme zu schaffen, die auch Quanteninformationen fehlertoleranter machen [28].

Darüber hinaus wird die steigende Nachfrage nach robusten und zuverlässigen Speichersystemen in Bereichen wie Cloud-Computing und Big Data voraussichtlich die Weiterentwicklung und Anwendung von Reed-Solomon-Codes beeinflussen. Die zunehmende Komplexität und Größe von Datensätzen erfordert fortschrittliche Fehlerkorrekturmechanismen, um die Integrität und Verfügbarkeit von großen Datenmengen zu gewährleisten [29, Kapitel 5].

## **6.3 Fazit**

Reed-Solomon-Codes haben sich seit ihrer Einführung im Jahr 1960 als eine der robustesten und effektivsten Methoden zur Fehlerkorrektur und -detektion etabliert. Ihre Anwendung reicht von der Weltraumkommunikation über optische Datenträger bis hin zu modernen Speicher- und Kommunikationssystemen [4]. Trotz des Fortschritts in der Technologie und der Entwicklung neuer Fehlerkorrekturverfahren bleiben Reed-Solomon-Codes aufgrund ihrer Zuverlässigkeit ein unverzichtbares Werkzeug in vielen Anwendungsbereichen.

Die kontinuierliche Forschung und Entwicklung in diesem Bereich verspricht, die Einsatzmöglichkeiten von Reed-Solomon-Codes zu erweitern und ihre Leistungsfähigkeit zu steigern [27], [30]. In einer zunehmend digitalisierten Welt, in der die Zuverlässigkeit und Integrität von Daten von großer Bedeutung sind, werden Reed-Solomon-Codes daher auch in Zukunft eine zentrale Rolle spielen.

# Anhang A

## Weiterführende Informationen

### A.1 Reed-Solomon mit BCH-Schema

Das Reed-Solomon-Verfahren basiert auf dem Bose-Chaudhuri-Hocquenghem (BCH)-Schema und arbeitet wie der ursprüngliche Reed-Solomon-Code im Galois-Feld  $GF(2^r)$ . Die Daten werden als Polynom  $p(x)$  über diesem Galois-Körper dargestellt. Um das Codewort zu erzeugen, wird dieses Nachrichtenpolynom  $p(x)$  mit einem Generatorpolynom  $g(x)$  multipliziert [31].

$$c(x) = p(x)g(x)$$

Dieses Generatorpolynom wird bei der Spezifizierung der Ausprägung des eingesetzten Verfahrens festgelegt und ist somit beim En- und Decodieren bekannt [32].

Das Codepolynom  $c(x)$  wird durch dessen Koeffizienten an den Empfänger übertragen oder auf ein Medium gespeichert. Zum Decodieren wird die Encodierungsgleichung nach  $p(x)$  aufgelöst und dann wird  $p(x) = \frac{c(x)}{g(x)}$  mit Hilfe von Polynomdivision bestimmt [5, Kapitel 1.4].

### A.2 Beweis des Hamming-Abstands von Reed-Solomon-Codes

Wie schon in Abschnitt 4.2 auf Seite 9 erwähnt, sind zum Decodieren eines Codeworts  $\binom{n}{k}$  Gleichungssysteme möglich. Davon führen  $\binom{n-t}{k}$  Gleichungssysteme zur richtigen Lösung.  $\binom{t+k-1}{k}$  Gleichungssysteme, die mindestens eine fehlerbehaftete Gleichung enthalten, ergeben jeweils falsche Lösungen. Von all diesen Lösungen wird diejenige ausgewählt, welche am häufigsten aufgetreten ist und damit korrekt ist. Damit diese Entscheidung erfolgreich die richtige Lösung auswählt, muss also  $\binom{n-t}{k} > \binom{t+k-1}{k}$  sein [3]. Durch Umformung entsteht folgende Ungleichung, die für  $n$ ,  $k$  und  $t$  erfüllt sein muss, damit das Reed-Solomon-Verfahren funktioniert.

$$\binom{n-t}{k} > \binom{t+k-1}{k}$$
$$n - k + 1 > 2t$$

Das bedeutet, dass es nie  $n - k + 1$  oder mehr Fehler geben darf [3]. Das entspricht dem Hamming-Abstand, der genau diese Obergrenze für ECCs angibt.

### A.3 Vergleich mit anderen Codierungsverfahren

|              | Codelänge<br><b>n</b> | Nachrichtenlänge<br><b>k</b> | Fehlererkennung<br><b>2t</b> | Fehlerkorrektur<br><b>t</b> |
|--------------|-----------------------|------------------------------|------------------------------|-----------------------------|
| Hamming      | 6                     | 3                            | 2                            | 1                           |
| Reed-Solomon | 5                     | 3                            | 2                            | 1                           |
| BCH          | 7                     | 3                            | 2                            | 1                           |
| Hamming      | 255                   | 247                          | 2                            | 14                          |
| Reed-Solomon | 255                   | 223                          | 32                           | 16                          |
| BCH          | 255                   | 179                          | 20                           | 10                          |

Tabelle A.1: Tabelle zum Vergleich verschiedener Codierungsverfahren

In Tabelle A.1 werden drei gängige Codierungsverfahren - Hamming, Reed-Solomon und BCH - hinsichtlich ihrer Parameter Codelänge  $n$ , Nachrichtenlänge  $k$ , Fehlererkennung  $2t$  und Fehlerkorrektur  $t$  verglichen.

Der erste Teil der Tabelle zeigt die Parameter beispielhaft für Nachrichten der Länge  $k = 3$ . Dazu wurde die Codelänge so gewählt, dass alle drei Verfahren die gleiche Anzahl an Fehlern korrigieren bzw. erkennen können. Dieses Beispiel ist nicht repräsentativ, da typischerweise längere Nachrichten verwendet werden. Es verdeutlicht aber mit wie viel Redundanz der gleiche Effekt bewirkt werden kann. Bei Reed-Solomon werden lediglich zwei Redundanzsymbole benötigt, beim Hamming-Code drei und bei BCH vier [33], also etwas mehr als beim Reed-Solomon-Verfahren.

Der zweite Teil der Tabelle zeigt die Parameter für längere Codelängen und zwar  $n = 255$ . Beim Hamming-Code ergibt sich daraus eine Nachrichtenlänge von 247. Bei den beiden anderen Verfahren wurde je eine in der Praxis typische Nachrichtenlänge gewählt [8, Kapitel 6.2]. Die Anzahl der Fehler ergibt sich durch Berechnung. Beim Hamming-Code gilt das Prinzip single error correction-double error detection (SEC-DED) und dadurch wird die Effektivität bei längeren Codes nicht besser. Dieser eignet sich daher nur für Anwendungsfälle mit geringer Fehleranfälligkeit [16]. Bei den anderen beiden Verfahren ist die Anzahl der Fehler, die korrigiert bzw. erkannt werden können, wesentlich höher [34]. Der Reed-Solomon-Code ist am effektivsten.

Die Wahl des Codierungsverfahrens hängt daher von den spezifischen Anforderungen der Anwendung ab, insbesondere von der benötigten Fehlerkorrekturfähigkeit und den aufzuwendenden Kosten [35].



## A.4 Beispielhafte Durchführung des Reed-Solomon-Verfahrens

Um auf eine gegebene Nachricht  $m = [2, 4, 3]$  das Reed-Solomon-Verfahren anzuwenden, wird im Folgenden der ursprüngliche Ansatz aus den Abschnitten 4.1 auf Seite 9 und 4.2 auf Seite 9 umgesetzt. Es wird  $RS(5, 3)$  verwendet, wodurch mit zwei Redundanzsymbolen ein Fehler korrigiert werden kann. Die Fehlererkennung wird hier nicht betrachtet. Die Operationen werden über dem Körper  $GF(5)$  mit der primitiven Einheitswurzel  $\alpha = 2$  ausgeführt.

Die Nachricht wird zum Encodieren in das Nachrichtenpolynom

$$p(x) = m_0 + m_1x + m_2x^2 = 2 + 4x + 3x^2$$

umgeformt. Um das Codewort zu generieren werden die Potenzen der Einheitswurzel  $\alpha$ , also alle Werte des Körpers  $GF(5) \setminus \{0\} = [1, 2, 3, 4]$  und 0, eingesetzt.

$$\begin{aligned} p(0) &= 2 && \equiv 2 \pmod{5} \\ p(1) &= 2 + 4 + 3 &= 9 &\equiv 4 \pmod{5} \\ p(2) &= 2 + 8 + 12 &= 22 &\equiv 2 \pmod{5} \\ p(3) &= 2 + 12 + 27 &= 41 &\equiv 1 \pmod{5} \\ p(4) &= 2 + 16 + 48 &= 66 &\equiv 1 \pmod{5} \end{aligned}$$

Somit entsteht  $c = [2, 4, 2, 1, 1]$ , welches nun im Übertragungskanal oder Speichermedium verfälscht werden kann.

Beim Empfänger kommt so beispielsweise  $\hat{c} = [2, 4, 3, 1, 1]$  und nicht  $c$  an. Zum Decodieren werden folgende Gleichungen aufgestellt:

$$\begin{aligned} c_0 &= 2 = p(0) = m_0 + m_1 \cdot 0 + m_2 \cdot 0^2 \\ c_1 &= 4 = p(1) = m_0 + m_1 \cdot 1 + m_2 \cdot 1^2 \\ c_2 &= 3 = p(2) = m_0 + m_1 \cdot 2 + m_2 \cdot 2^2 \\ c_3 &= 1 = p(3) = m_0 + m_1 \cdot 3 + m_2 \cdot 3^2 \\ c_4 &= 1 = p(4) = m_0 + m_1 \cdot 4 + m_2 \cdot 4^2 \end{aligned}$$

Davon ist möglicherweise eine falsch. Es ist aber nicht bekannt, ob dies der Fall ist oder welche der Gleichungen betroffen ist. Es sind drei Unbekannte, die Symbole der Nachricht, gesucht.

Deshalb werden jeweils drei Gleichungen zu einem Gleichungssystem zusammen gefasst und gelöst. Beispielfhaft ergibt das Gleichungssystem der ersten drei Gleichungen

$$m_0 + 0m_1 + 0m_2 = 2$$

$$m_0 + 1m_1 + 1m_2 = 4$$

$$m_0 + 2m_1 + 4m_2 = 3$$

mit Hilfe eines geeigneten Lösungsverfahrens  $m_0 = 2$ ;  $m_1 = 1$ ;  $m_2 = 1$ . Das Gleichungssystem der ersten beiden und der vierten Gleichung hat die Lösung  $m_0 = 2$ ;  $m_1 = 4$ ;  $m_2 = 3$ . Diese Lösung wird für alle  $\binom{5}{3} = 10$  möglichen Gleichungssysteme bestimmt:

$$(c_0; c_1; c_2) \rightarrow (2; 1; 1)$$

$$(c_0; c_1; c_3) \rightarrow (2; 4; 3)$$

$$(c_0; c_1; c_4) \rightarrow (2; 4; 3)$$

$$(c_0; c_2; c_3) \rightarrow (2; 3; 4)$$

$$(c_0; c_2; c_4) \rightarrow (2; 0; 4)$$

$$(c_0; c_3; c_4) \rightarrow (2; 4; 3)$$

$$(c_1; c_2; c_3) \rightarrow (4; 3; 2)$$

$$(c_1; c_2; c_4) \rightarrow (0; 4; 0)$$

$$(c_1; c_3; c_4) \rightarrow (2; 4; 3)$$

$$(c_2; c_3; c_4) \rightarrow (3; 3; 1)$$

Man erkennt, dass viele verschiedene Lösungen herauskommen. Jedoch die Lösung

$$m_0 = 2; m_1 = 4; m_2 = 3$$

vier mal. Alle anderen Ergebnisse kommen jeweils nur einmal vor. So weiß man, dass  $m = [2, 4, 3]$  die ursprüngliche Nachricht war.

## A.5 Umsetzungsparameter der verschieden Anwendungsfälle

**Umsetzungsparameter Voyager [8, Kapitel 5]:**

Code-Wort-Länge: 255 Symbole; Daten-Symbole: 223; Redundanz-Symbole: 32

**Umsetzungsparameter DVB-H [22]:**

Code-Wort-Länge: 255 Symbole; Daten-Symbole: 191; Redundanz-Symbole: 64

**Umsetzungsparameter für CDs [4, Kapitel 4]:**

Code-Wort-Länge: 32 Symbole; Daten-Symbole: 28; Redundanz-Symbole: 4

**Umsetzungsparameter für DVD und Blu-Ray [9]:**

Doppeltes En- bzw. Decoding

Innerer Code: Code-Wort-Länge: 182 Symbole; Daten-Symbole: 172; Redundanz-Symbole: 10

Äußerer Code: Code-Wort-Länge: 208 Symbole; Daten-Symbole: 192; Redundanz-Symbole: 16

**Umsetzungsparameter für RAID6-Systeme [23]:**

Code-Wort-Länge: Abhängig von der Konfiguration; Typische Redundanz: 3 Datenlaufwerke und 2 Redundanzlaufwerke

**Umsetzungsparameter für QR-Codes [26]:**

Verschiedene Fehlerkorrekturstufen:

L (7% der Daten können korrigiert werden), M (15%), Q (25%), H (30%)

**Umsetzungsparameter für QR-Codes Stufe Low [26]:**

Code-Wort-Länge: 26 Symbole; Daten-Symbole: 19; Redundanz-Symbole: 7

## A.6 Fehlertoleranz bei QR-Codes

Quick Response Codes (QR-Codes) sind zweidimensionale Barcodes, die, dank der Integration von Reed-Solomon-Codes, eine hohe Fehlertoleranz aufweisen. Diese Fehlerkorrekturmethode ermöglichen das Auslesen von QR-Codes selbst bei teilweise beschädigten oder verschmutzten Codes.

Für QR-Codes existieren vier verschiedene Stufen der Fehlerkorrektur, die abhängig von der zu codierenden Datenmenge ausgewählt werden können. Die vier Stufen sind: L (Low); M (Medium); Q (Quartile); H (High) So können je nach Stufe 7% bis 30% der Daten wiederhergestellt werden [25, Kapitel 3].

Anhand der Stufe L wird nun die konkrete Umsetzung dargestellt. Die Struktur von QR-Codes ist so aufgebaut, dass die Quadrate, auch Module genannt, nach einem festgelegtem Schema angeordnet sind. Jedes Modul repräsentiert ein Bit (Weiß entspricht 0; Schwarz entspricht 1). Einige sind in jedem QR-Code gleich. Andere stellen die Metadaten zur Verfügung. Die Übrigen codieren die eigentlichen Nutzdaten [25, Kapitel 3]. Die Metadaten beinhalten die Umsetzungsparameter zur Fehlerkorrektur (siehe Abbildung A.1 auf Seite 23). Die Nutzdaten sind, wie in

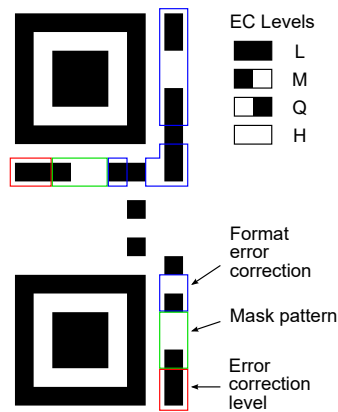


Abbildung A.1: Metadaten eines QR-Codes (Stufe L) [26]

Abbildung A.2 auf Seite 23 zu sehen, als Symbole mit je acht Modulen codiert. Es können insgesamt 26 Symbole im QR-Code platziert werden. Bei dem eingesetzten Reed-Solomon-Code  $RS(255, 248)$  gekürzt auf  $RS(26, 19)$  sind 19 Symbole für die Nutzdaten und sieben Symbole  $E1$  bis  $E7$  für die Fehlertoleranz. Da zur Überprüfung nochmals zwei Symbole der Nachricht als Metadaten gebraucht werden, bleiben nur 17 der 19 Symbole für die eigentliche Nachricht [36]. Damit können bis zu 2 Symbole also 16 Module korrigiert werden [26].

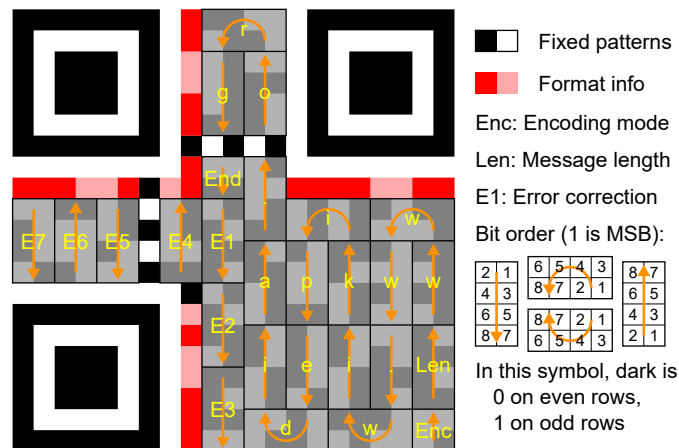


Abbildung A.2: Datensymbole in einem QR-Code (Stufe L) [26]

Diese Fehlerkorrekturmechanismen verleihen QR-Codes Robustheit gegenüber physischen Beschädigungen wie Rissen, Kratzern oder Verunreinigungen. Die Fähigkeit, selbst unter solchen herausfordernden Bedingungen die gespeicherten Daten korrekt wiederherzustellen, hat zur weitverbreiteten Nutzung von QR-Codes in verschiedenen Bereichen geführt, darunter Einzelhandel, Logistik und Zugangskontrollen [26].

# Glossar

**ARQ** „automatic repeat request“. i, 3

**ATSC** Advanced Television Systems Committee. i, 13

**BCH** Bose-Chaudhuri-Hocquenghem. i, 3, 8, 10, 14, 18, 19

**CD** Compact Disc. i, 1, 4, 14, 22

**DSL** Digital Subscriber Line. i, 14

**DVB** Digital Video Broadcasting. i, 14

**DVB-H** Digital Video Broadcasting-Handheld. i, 14, 21

**DVD** Digital Video Disc. i, 14, 22

**ECC** Fehlerkorrektur-Code. i, iv, 3, 4, 19

**FEC** Vorwärtskorrektur-Verfahren. i, 3

**MIT** Massachusetts Institute of Technology. i, 3

**QR-Code** Quick Response Code. i, 1, 14, 15, 22, 23

**RAID** Redundant Array of Independent Disks. i, 14, 22

**SEC-DED** single error correction-double error detection. i, 19

**WiMAX** Worldwide Interoperability for Microwave Access. i, 14

# Literatur

- [1] W. W. Peterson und E. J. Weldon, *Error-Correcting Codes*. MIT Press, 1972, ISBN: 978-0-262-16039-1.
- [2] B. Friedrichs, *Kanalcodierung*. Berlin, Heidelberg: Springer, 1996, ISBN: 978-3-540-59353-9 978-3-642-60985-5. DOI: 10.1007/978-3-642-60985-5.
- [3] I. S. Reed und G. Solomon, „Polynomial Codes over Certain Finite Fields“, *Journal of the society for industrial and applied mathematics*, Jg. 8, Nr. 2, S. 300–304, 1960.
- [4] S. B. Wicker, Hrsg., *Reed Solomon Codes and Their Applications*. Piscataway, NJ: IEEE Press, 1994, ISBN: 978-0-7803-5391-6.
- [5] W. A. Geisel, *Tutorial on Reed-Solomon Error Correction Coding*, Aug. 1990.
- [6] E. Berlekamp, „Nonbinary BCH Decoding (Abstr.)“, *IEEE Transactions on Information Theory*, Jg. 14, Nr. 2, S. 242–242, März 1968, ISSN: 1557-9654. DOI: 10.1109/TIT.1968.1054109.
- [7] J. Massey, „Shift-Register Synthesis and BCH Decoding“, *IEEE Transactions on Information Theory*, Jg. 15, Nr. 1, S. 122–127, Jan. 1969, ISSN: 1557-9654. DOI: 10.1109/TIT.1969.1054260.
- [8] R. Ludwig und J. Taylor, *Voyager Telecommunications*, März 2002.
- [9] H. Chang und C. Shung, „A Reed-Solomon Product-Code (RS-PC) Decoder for DVD Applications“, in *1998 IEEE International Solid-State Circuits Conference. Digest of Technical Papers, ISSCC. First Edition (Cat. No.98CH36156)*, Feb. 1998, S. 390–391. DOI: 10.1109/ISSCC.1998.672548.
- [10] J. Wendling, *Introduction to Reed-Solomon*, <https://innovation.vivint.com/introduction-to-reed-solomon-bc264d0794f8>, Aug. 2017. (besucht am 12.05.2024).
- [11] E. Weitz, *Konkrete Mathematik (nicht nur) für Informatiker: Mit vielen Grafiken und Algorithmen in Python*. Berlin, Heidelberg: Springer, 2021, ISBN: 978-3-662-62617-7 978-3-662-62618-4. DOI: 10.1007/978-3-662-62618-4.
- [12] C. Schulz-Hanke, „BCH codes with combined error correction and detection BCH Codes mit kombinierter Korrektur und Erkennung“, Diss., Universität Potsdam, 2023, 5094 KB, ii, 191 pages. DOI: 10.25932/PUBLISHUP-61794.

- [13] M. Riley und I. Richardson, *Reed-Solomon Codes*, [https://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed\\_solomon\\_codes.html](https://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed_solomon_codes.html). (besucht am 20.05.2024).
- [14] B. Hubert, *Practical Reed-Solomon for Programmers*, <https://berthub.eu/articles/posts/reed-solomon-for-programmers/>, Juni 2021. (besucht am 12.05.2024).
- [15] T. Verbeure, *Reed-Solomon Error Correcting Codes from the Bottom Up*, <https://tomverbeure.github.io/2022/08/07/Reed-Solomon.html>, Aug. 2022. (besucht am 12.05.2024).
- [16] L. Williams, *Hamming-Code: Fehlererkennung und -korrektur mit Beispielen*, <https://www.guru99.com/de/hamming-code-error-correction-example.html>, Feb. 2024. (besucht am 24.06.2024).
- [17] G. Feng, *The Berlekamp-Welch Algorithm: A Guide*.
- [18] J. Biernat, T. Serafin und W. Kulikowski, „Hardware Implementation of the Reed-Solomon Decoder“, in *2010 IEEE 14th International Conference on Intelligent Engineering Systems*, Mai 2010, S. 255–257. DOI: 10.1109/INES.2010.5483838.
- [19] S. Southwell, *Introduction to Error Detection and Correction #2: Reed-Solomon*, <https://www.linkedin.com/pulse/introduction-error-detection-correction-2-simon-southwell>. (besucht am 20.05.2024).
- [20] *Was ist das Reed-Solomon-Verfahren?*, <https://www.egovernment.de/was-ist-das-reed-solomon-verfahren-a-cbb3a5d0b405888012d37555a7ef5225/>, Nov. 2022. (besucht am 20.05.2024).
- [21] T. Iliev, I. Lokshina, D. Radev und G. Hristov, „Analysis and Evaluation of Reed-Solomon Codes in Digital Video Broadcasting Systems“, in *2008 Wireless Telecommunications Symposium*, Apr. 2008, S. 92–96. DOI: 10.1109/WTS.2008.4547549.
- [22] „DVB-H“, *Wikipedia*, Jan. 2024. (besucht am 20.05.2024).
- [23] *RAID 6: Storage technology to minimize data loss*, <https://www.ionos.com/digitalguide/server/security/raid-6/>, Aug. 2021. (besucht am 20.05.2024).
- [24] W. C. Vermillion, *End-to-End DSL Architectures*. Cisco Press, Apr. 2003, ISBN: 978-1-58705-087-9.
- [25] S. Tiwari, „An Introduction to QR Code Technology“, in *2016 International Conference on Information Technology (ICIT)*, Dez. 2016, S. 39–44, ISBN: 978-1-5090-3584-7. DOI: 10.1109/ICIT.2016.021.
- [26] „QR code“, *Wikipedia*, Mai 2024. (besucht am 20.05.2024).

- [27] R. Con, A. Shpilka und I. Tamo, „Optimal Two-Dimensional Reed–Solomon Codes Correcting Insertions and Deletions“, *IEEE Transactions on Information Theory*, Jg. 70, Nr. 7, S. 5012–5016, Juli 2024, ISSN: 1557-9654. DOI: 10.1109/TIT.2024.3387848.
- [28] M. Grassl, W. Geiselmann und T. Beth, „Quantum Reed-Solomon Codes“, Jg. 1719, S. 231–244, 1999. DOI: 10.1007/3-540-46796-3\_23. arXiv: quant-ph/9910059.
- [29] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos et al., „XORing Elephants: Novel Erasure Codes for Big Data“, *Proceedings of the VLDB Endowment*, Jg. 6, Nr. 5, S. 325–336, März 2013, ISSN: 2150-8097. DOI: 10.14778/2535573.2488339.
- [30] C. Sippel, C. Ott, S. Puchinger und M. Bossert, „Reed–Solomon Codes over Fields of Characteristic Zero“, in *2019 IEEE International Symposium on Information Theory (ISIT)*, Juli 2019, S. 1537–1541. DOI: 10.1109/ISIT.2019.8849332.
- [31] J. J. Stone, „Multiple-Burst Error Correction with the Chinese Remainder Theorem“, *Journal of the Society for Industrial and Applied Mathematics*, Jg. 11, Nr. 1, S. 74–81, März 1963, ISSN: 0368-4245. DOI: 10.1137/0111005.
- [32] deepak2018, *What is Reed–Solomon Code?*, <https://www.geeksforgeeks.org/what-is-reed-solomon-code/>, Feb. 2022. (besucht am 20.05.2024).
- [33] „BCH-Code“, *Wikipedia*, Dez. 2023. (besucht am 20.05.2024).
- [34] D. L. Neuhoff, *Digital Communications Signals and Systems*.
- [35] D. Abrha, M. Frdiesa und M. Wutabachew, „Comparison of hamming, BCH, and reed Solomon codes for error correction and detecting techniques“, *International Journal of Engineering Trends and Technology*, Jg. 67, Nr. 6, S. 1–4, Juni 2019, ISSN: 22315381. DOI: 10.14445/22315381/IJETT-V67I6P201.
- [36] Pillazo, *How to Decode a QR Code by Hand*, März 2013. (besucht am 20.05.2024).