

Duale Hochschule Baden-Württemberg Mannheim

## **Studienarbeit**

# **Konstruktion eines kryptographischen Verfahrens basierend auf Reed-Solomon-Codes und Diskussion möglicher Angriffsmethoden**

**Studiengang Informatik**

**Studienrichtung Cyber Security**

Verfasser(in):	Roman Wetenkamp
Matrikelnummer:	5533869
Kurs:	TINF20CS1
Studiengangsleiter:	Prof. Dr. Konstantin Bayreuther
Wissenschaftliche(r) Betreuer(in):	Prof. Dr. Reinhold Hübl
Bearbeitungszeitraum:	18.10.2022 – 18.04.2023
Version:	23. Januar 2023

# Ehrenwörtliche Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit mit dem Titel "*Konstruktion eines kryptographischen Verfahrens basierend auf Reed-Solomon-Codes und Diskussion möglicher Angriffsmethoden*" selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum

Roman Wetenkamp

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>iii</b>
<b>Quelltextverzeichnis</b>	<b>iv</b>
<b>Abkürzungsverzeichnis</b>	<b>v</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Thematische Übersicht . . . . .	2
<b>2 Codierungstheorie</b>	<b>3</b>
2.1 Übermittlung von Informationen . . . . .	3
2.2 Problemstellung und Zielsetzung . . . . .	5
2.3 Kanalcodierung . . . . .	5
2.3.1 Grundbegriffe . . . . .	6
2.3.2 Noisy Channels Coding Theorem . . . . .	9
2.4 Finitfeldarithmetik . . . . .	10
2.4.1 Gruppen . . . . .	10
2.4.2 Körper . . . . .	12
2.4.3 Polynome . . . . .	15
<b>3 Lineare fehlerkorrigierende Codes für kryptographische Zwecke</b>	<b>17</b>
3.1 Zyklische Codes . . . . .	19
3.2 Definition der Klasse von Reed-Solomon-Codes . . . . .	21
3.2.1 Verallgemeinerte Reed-Solomon-Codes . . . . .	22
3.2.2 Duale Codes . . . . .	23
3.2.3 Kontroll- und Generatormatrizen . . . . .	24
3.3 QR-Codes als Anwendungsbeispiel . . . . .	25
3.3.1 Hintergrund und Konzept . . . . .	26
3.3.2 Konstruktion von Fehlerkorrekturworten . . . . .	26
3.3.3 Dekodierung . . . . .	27
3.4 Algorithmische Betrachtung . . . . .	29
3.4.1 Kodierung von Reed-Solomon-Codes . . . . .	29
3.4.2 Fehlerkorrektur und Dekodierung . . . . .	32
<b>Literaturverzeichnis</b>	<b>35</b>

# Abbildungsverzeichnis

1.1	Fortlaufend ergänzte Übersicht über relevante und abgegrenzte Teilgebiete der Informationstheorie . . . . .	2
2.1	Gegenstand der Codierungstheorie (nach [7, S. 1]) . . . . .	5
2.2	Vereinfachte Darstellung eines Information Transmission System (ITS) (nach [4, S. 3]) . . . . .	6
2.3	Visualisierung der Kugelinterpretation (nach [9]) . . . . .	9
3.1	Anordnung einzelner Datencodeworte und Fehlerkorrekturcodeworte in einem QR-Code [23, S. 48] . . . . .	27

# Quelltextverzeichnis

3.1 SageMath-Implementierung des Kodieralgorithmus . . . . .	30
--	----

# Abkürzungsverzeichnis

<b>BSC</b>	Binary Symmetric Channel
<b>EDV</b>	Elektronische Datenverarbeitung
<b>ITS</b>	Information Transmission System
<b>QR</b>	Quick Response
<b>RSC</b>	Reed-Solomon-Codes

# 1 Einleitung

„The lesson here is that it is insufficient to protect ourselves with laws;  
we need to protect ourselves with mathematics.“  
– BRUCE SCHNEIER in [1]

In einer Welt, in der so viele Daten wie nie zuvor übertragen werden, digitale Kriegsführung und *Nation-state-attacks* nicht mehr bloß Gegenstand dystopischer Science-Fiction-Literatur, sondern Alltag sind, steigt die Relevanz kryptographischer Verfahren, die es ermöglichen, die Vertraulichkeit und Integrität schützenswerter Daten selbst unter der Annahme, dass Angreifenden nahezu unbegrenzte Ressourcen zur Verfügung stehen, sicherzustellen.

Das Forschungsgebiet der *Post-Quanten-Kryptographie* [vgl. 2] hat die Entwicklung kryptographischer Systeme zum Gegenstand, die selbst mit den durch Quantentechnologie anzunehmenden Rechenleistungssteigerungen nicht gebrochen werden können. Ein aussichtsreicher Kandidat dafür ist das *McEliece*-Kryptosystem, das auf linearen, fehlerkorrigierenden Codes basiert. Jene Verbindung der Codierungstheorie und Kryptographie ist Gegenstand dieser Arbeit: Basierend auf dem *McEliece*-Kryptosystem soll ein aufbauender Ansatz von HARALD NIEDERREITER betrachtet werden, der im Vergleich zum *McEliece*-Kryptosystem nicht auf *Goppa*-, sondern auf *Reed-Solomon*-Codes basiert und dadurch zwar bessere Rechenzeiten erreicht, jedoch vermutlich auch an Sicherheit einbüßt.

Diese Arbeit stellt zunächst die theoretischen Hintergründe der Codierungstheorie für kryptographische Zwecke dar, bevor basierend darauf die Arbeiten von MCELIECE und NIEDERREITER analysiert und für die Entwicklung eines eigenen Kryptosystems genutzt werden. Auf jenes Verfahren werden abschließend Methoden der Kryptoanalyse unter Einbeziehung der Arbeit von SIDELNIKOV und SHESTAKOV angewandt, um Aussagen über die Sicherheit des Verfahrens treffen zu können.

## 1.1 Thematische Übersicht

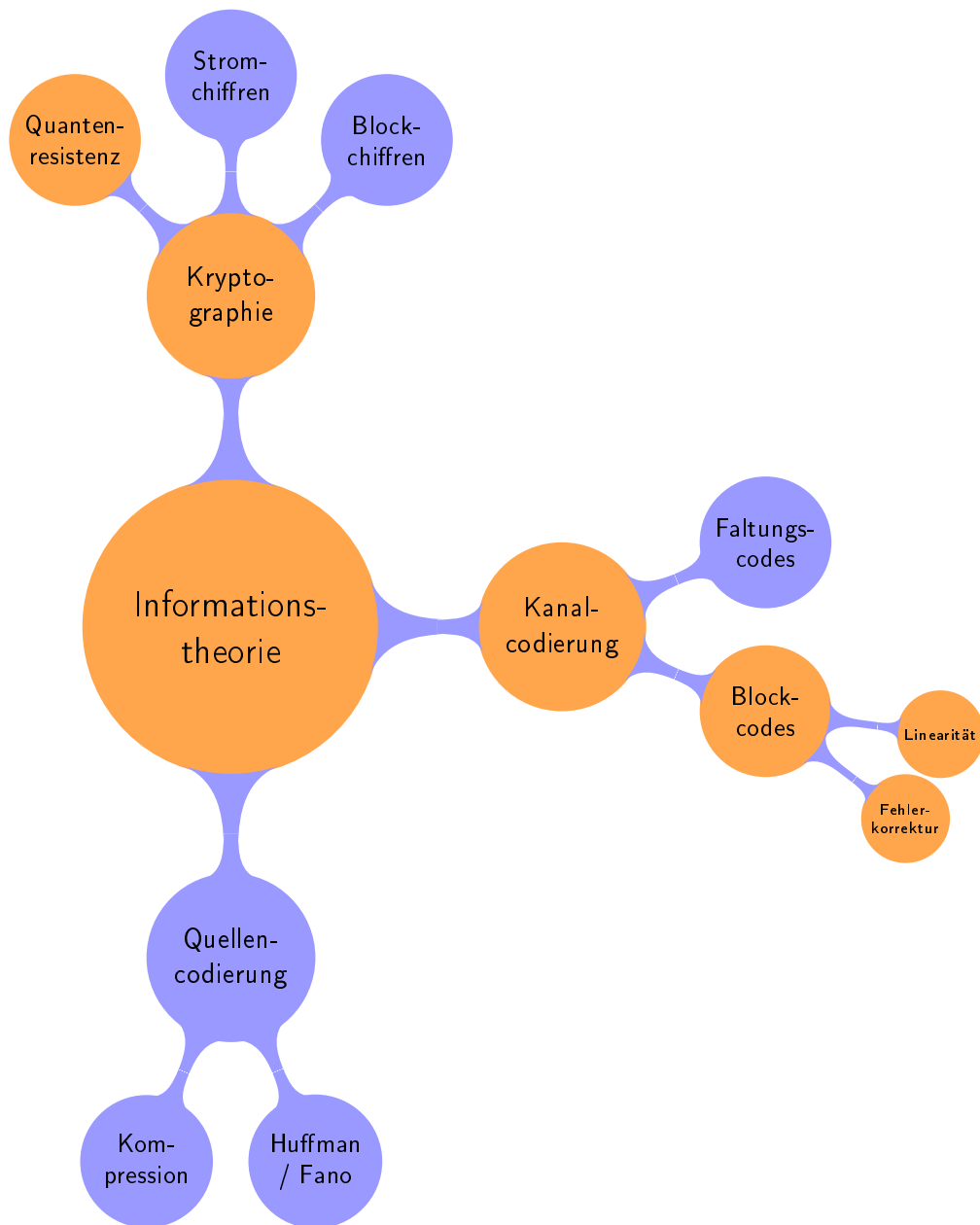


Abbildung 1.1: Fortlaufend ergänzte Übersicht über relevante und abgegrenzte Teilgebiete der Informationstheorie



## 2 Codierungstheorie

Da im Rahmen dieser Arbeit ein kryptographisches Verfahren entwickelt wird, das auf Elementen der Codierungstheorie basiert, wird diese nun zunächst in Definitionen und Hintergründen dargestellt.

### 2.1 Übermittlung von Informationen

Sowohl in der Kryptographie, als auch in der Codierungstheorie fungieren **Nachrichten**, die über mit bestimmten Eigenschaften behaftete **Kanäle** übertragen werden, als die Objekte der Anschauung.

#### Definition 1

Eine **Nachricht**  $m$  sei definiert als eine endliche Folge von Zeichen  $a_i \in \Sigma$ , wobei  $\Sigma$  eine endliche Menge von Zeichen (genannt **Alphabet**) bezeichnet.

$$m = \langle a_1, a_2, \dots, a_{n-1}, a_n \rangle \quad \forall i = 1, \dots, n : a_i \in \Sigma$$

Ein typisches Alphabet sind die Zeichen der *ASCII*-Kodierung, mit denen nahezu alle Worte und Sätze der natürlichen englischen Sprache gebildet werden können [vgl. 3]. Dieses Alphabet besteht nun nicht aus Zeichen der natürlichen Sprache, sondern aus 7 Bit langen Zahlenwerten, was die Anwendung von Codes oder kryptographischen Verfahren ermöglicht. Im Rahmen dieser Arbeit wird angenommen, dass Zeichen stets in einem Zahlenformat repräsentiert werden.

Die Definition einer informationstheoretischen Nachricht impliziert eine Autorenschaft, folglich muss jeder Nachricht eine Partei (ein natürliche Person, ein System oder ein Dienst) zugeordnet werden können, die im Folgenden als **Sender**<sup>1</sup> der Nachricht bezeichnet wird. Wird diese Nachricht nun über einen Kanal an eine andere Partei übertragen, so nennen wir

---

<sup>1</sup>Da sich die Anwendung der modernen Kryptographie sehr überwiegend mit dem Austausch von verschlüsselten Nachrichten zwischen Systemen und nicht unmittelbar zwischen natürlichen Personen befasst, wird hier die männliche Form verwendet (Sender = Dienst/System).

diese den **Empfänger**. Entgegen der in der Kryptographie üblichen *Alice-Bob*-Notation wird in dieser Arbeit diese Terminologie beibehalten, um an den codierungstheoretischen Hintergrund anzuknüpfen.

Ein **Kanal** bezeichne ein Medium zur Datenübertragung wie beispielsweise einen elektrischen Leiter, einen Lichtwellenleiter oder die Luft für eine drahtlose Verbindung. Es gibt Kanäle, die Informationen **digital** übertragen, also als diskrete Binärwerte, und im Gegensatz dazu Kanäle, die fortlaufend und stetig Signale übertragen [vgl. 4, S. 1]. **Rauschen** bezeichne nicht-deterministische Daten, die Nachrichten bei einer Übertragung über einen Kanal unbeabsichtigt hinzugefügt werden und so die Nachricht verändern, sie folglich mit **Fehlern** versetzen [vgl. 5, S. 1].

### Definition 2

Ein **Kanal**  $K$  mit der Menge der Eingabewerte  $I$  und der Menge der Ausgabewerte  $J$  sei definiert als Matrix

$$K_{ij} = \Pr(j \mid i) \quad i \in I, j \in J$$

wobei  $\Pr(j \mid i)$  definiert sei als die Wahrscheinlichkeit für die Ausgabe  $j$  unter der Bedingung  $i$ .

Sofern mindestens ein Koeffizient der Matrix  $i \neq j$  einen Wert  $> 0$  aufweist, ist der Kanal **rauschend** [vgl. 6, S. 73f.].

### Beispiel 1

Ein **Binary Symmetric Channel (BSC)**  $\Gamma$  mit einer Bitfehlerwahrscheinlichkeit  $e$  ist definiert durch

$$\Gamma = \begin{pmatrix} \Gamma_{00} & \Gamma_{01} \\ \Gamma_{10} & \Gamma_{11} \end{pmatrix} = \begin{pmatrix} 1-e & e \\ e & 1-e \end{pmatrix}$$

Dass ein Kanal eine Nachricht ohne Rauschen überträgt, ist zwar ein erstrebenswerter Zustand, praktisch jedoch aufgrund der Physik nicht zu erreichen. Jede physische Datenübertragung verläuft nicht fehlerfrei, weshalb die Beziehung  $m = m'$  daher nur in einem theoretischen Idealfall gilt. Diese Feststellung liefert die Begründung für die Beschäftigung mit der Codierungstheorie.

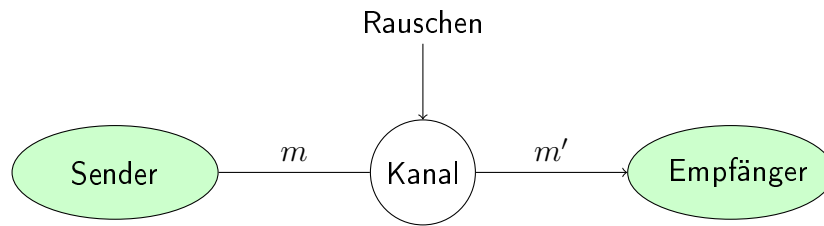


Abbildung 2.1: Gegenstand der Codierungstheorie (nach [7, S. 1])

## 2.2 Problemstellung und Zielsetzung

Da die Datenübertragung über eine Vielzahl von Kanälen eben nicht fehlerfrei verläuft, liegt es nahe, die Daten so zu übertragen, dass fehlende Bits aus dem Rest der Nachricht erschlossen werden können, wie es zum Beispiel bei natürlicher Sprache der Fall ist. Unsere Worte enthalten häufig Buchstaben, die nicht zwingend erforderlich sind, um das gemeinte Wort zu erkennen [vgl. 5, S. 3].

Überträgt man diese Erkenntnis auf Nachrichten einer beliebigen Sprache, so lassen sich auch im allgemeinen Fall durch das Hinzufügen von redundanten Informationen Nachrichten erzeugen, deren Informationsgehalt sich auch nach der Übermittlung nicht verringert hat. Ein spezieller Typ dieser Verfahren wird als **fehlerkorrigierende Codes** bezeichnet [vgl. 5, S. 3]. Diese Codes sind dem Gebiet der **Kanalcodierung** zuzurechnen, die das Ziel hat, die Qualität der Übertragung auf verlustbehafteten Kanälen sicherzustellen. Sie grenzt sich ab von der **Quellencodierung**, die Verfahren bündelt, welche die Transformation der zu versendenden Daten zum Ziel hat, beispielsweise zur Kompression und Redundanzverringerung [vgl. 8, S. 1]. Der Fokus dieser Arbeit liegt dabei auf der Kanalcodierung, da die betrachteten kryptographischen Verfahren auf ihr basieren.

## 2.3 Kanalcodierung

Die Relevanz der Frage, wie möglichst viele Übertragungsfehler in einer Datenübertragung vermieden oder korrigiert werden können, stieg mit der Verbreitung von EDV-Systemen und nicht zuletzt dem Internet rasant an [vgl. 4, S. 209]. Populäre Beiträge der Grundlagenforschung wie jener von HAMMING sind daher auch trotz ihres einige Dekaden umspannenden Alters Fundament der folgenden Definitionen.

### 2.3.1 Grundbegriffe

Um Codier- und Decodiervorgänge beschreiben zu können, ist eine Erweiterung und Präzisierung der Abbildung 2.1 erforderlich. Die Zuordnung eines Codewortes  $c$  zu einer Nachricht  $m$  wird als **Codierung** (der Nachricht) bezeichnet.

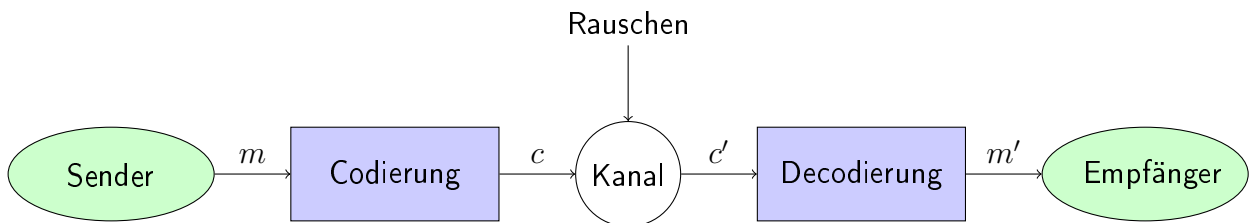


Abbildung 2.2: Vereinfachte Darstellung eines ITS (nach [4, S. 3])

#### Definition 3

Sei  $A$  ein Alphabet und  $n \in \mathbb{N}$ .

Dann sei  $A^n$  die Menge aller  $n$ -Tupel der Form  $A^n = \{\langle a_1, a_2, \dots, a_n \rangle \mid a_i \in A\}$ .

Ein **Blockcode**  $C$  der Länge  $n$  über dem Alphabet  $A$  ist definiert als  $C \subseteq A^n$ , wobei  $|C| > 0$  gelten muss.

Ist  $m = |C|$  und  $B$  mit  $|B| < m$  die Menge zu codierender Informationseinheiten über einem Alphabet  $A'$ , so ist jede injektive Abbildung  $f: B \rightarrow C$  eine **Codierfunktion** [vgl. 8, S. 10].

Neben Blockcodes können auch sogenannte **Faltungscodes** zur Fehlerkorrektur verwendet werden. Hierbei werden die Informationen nicht unabhängig voneinander blockweise codiert, sondern über Schieberegister in Abhängigkeit zueinander, wodurch sich eine bessere Performanz im Gegensatz zu Blockcodes ergibt [vgl. 10, S. 752]. Der Ansatz, Daten blockweise oder als Datenstrom über Schieberegister zu codieren, weist Parallelen zur kryptographischen Unterscheidung zwischen Block- und Stromchiffren auf. Der Fokus dieser Arbeit wird aufgrund der kryptographischen Bedeutung auf Blockcodes liegen.

Als Maß der Qualität einer Übertragung beziehungsweise der Verfälschung einer Nachricht durch Rauschen, aber auch für die Unterscheidbarkeit von Codeworten, bietet sich die **Hamming-Distanz** an.

**Definition 4**

Seien  $a = \langle a_1, a_2, \dots, a_n \rangle$ ,  $b = \langle b_1, b_2, \dots, b_n \rangle \in A^n$  und  $A$  ein Alphabet, so ist die **Hamming-Distanz**  $d$  eine Metrik auf  $A^n$ . Sie ist definiert als die Anzahl der abweichenden Stellen in zwei Codeworten  $a$  und  $b$ :

$$d(a, b) = |\{i \mid 1 \leq i \leq n, a_i \neq b_i\}|$$

Des Weiteren sei die **Minimaldistanz** eines Blockcodes  $C$  mit  $|C| > 1$  definiert als

$$d(C) = \min\{d(x, y) \mid x, y \in C, x \neq y\}$$

[vgl. 8, S. 11] [vgl. 11, S. 105] [vgl. 9, S. 155].

**Beispiel 2**

Seien  $A = \{0, 1\}$  und  $n = 5$ . Dann ist  $A^n = \{\langle 0, 0, 0, 0, 0 \rangle, \langle 0, 0, 0, 0, 1 \rangle, \dots, \langle 1, 1, 1, 1, 1 \rangle\}$ . Dann gilt für einen Blockcode  $C$ :

- $C = A^n \rightarrow d(C) = 1$
- $C = \{\langle 0, 0, 0, 0, 1 \rangle, \langle 0, 0, 0, 1, 0 \rangle, \dots, \langle 1, 0, 0, 0, 0 \rangle\} \rightarrow d(C) = 2$
- $C = \{\langle 0, 1, 0, 1, 0 \rangle, \langle 1, 0, 1, 0, 1 \rangle\} \rightarrow d(C) = 5$

Wenn die Werte einer Codierungsfunktion im Bildbereich weit verstreut sind, der Code also eine große Minimaldistanz aufweist, werden die einzelnen Codeworte unterscheidbarer. Folglich ergibt sich aus der Minimaldistanz ein relevantes Kriterium für die Eigenschaft eines Codes, **fehlererkennend** oder sogar **fehlerkorrigierend** zu sein. Sie beeinflusst, wie viele Fehler erkannt beziehungsweise korrigiert werden können [vgl. 9, S. 155].

HAMMING nutzt für seine Argumentation in [9] eine geometrische Betrachtung mithilfe von Kugeln:

**Definition 5**

Eine **abgeschlossene Kugel**  $B(a, r)$  um einen Punkt  $a$  mit Radius  $r$  in einem (metrischen) Raum  $X$  sei definiert durch

$$B(a, r) = \{x \in X \mid d(a, x) \leq r\}$$

Als Abstandsmaß  $d$  kann hier beispielsweise die eingeführte *Hamming*-Distanz verwendet werden.

### Definition 6

Sei  $B$  eine Kugel mit Radius  $r \geq 2$  und Mittelpunkt  $x \in A^n$ . Dann liegen alle  $a_i \in A^n$  mit  $d(a_i, x) = r$  auf der Kugeloberfläche von  $B$ . Sei  $C \subseteq A^n$  nun ein Blockcode mit  $d(C) = r$  und  $x \in C$ . Dann liegen alle  $c \in (C \setminus \{x\})$  auf der Kugeloberfläche oder außerhalb der Kugel  $B$ .

### Bemerkung 1

Ein Punkt  $a_j \neq x$  mit  $d(a_j, x) < r$  (unterscheidet sich in weniger als  $r$  Stellen von  $x$ ) liegt nicht auf der Kugeloberfläche und kann folglich kein fehlerfreies Codewort sein. Ein Code  $C$  mit Minimaldistanz 2 ist damit **fehlererkennend** für maximal ein falsch übertragenes Zeichen (notiert: „1-fehlererkennend“).

Ferner ist ein Code  $C'$  mit Minimaldistanz 3 **fehlerkorrigierend**: Sei  $a_r$  ein gültiges Codewort mit einer Minimaldistanz von 3 zu allen anderen Codeworten in  $C'$ . Für ein in einer Stelle abweichendes Wort  $a_f$  gilt folglich  $d(a_r, a_f) = 1$ , aber  $\forall a_i \in C' \setminus \{a_r, a_f\} : d(a_i, a_f) > 1$ . Damit lässt sich  $a_f$  eindeutig  $a_r$  zuordnen, wodurch sich der Fehler korrigieren lässt und  $C'$  **1-fehlerkorrigierend** ist [vgl. 9, S. 155f.].

Die Eigenschaft eines Codes, fehlererkennend oder fehlerkorrigierend zu sein, ist skalierbar:

### Definition 7

Seien  $c, c' \in C; c \neq c'$  Codeworte eines Blockcodes  $C \in A^n$ ,  $\epsilon \in \mathbb{N}$ ,  $B_\epsilon(c) = \{x \in A^n \mid d(x, c) \leq \epsilon\}$  und  $B_\epsilon(c')$  analog  $B_\epsilon(c') = \{x \in A^n \mid d(x, c') \leq \epsilon\}$ .

Dann ist  $C$   **$\epsilon$ -fehlerkorrigierend**, wenn  $\forall c, c' \in C : B_\epsilon(c) \cap B_\epsilon(c') = \emptyset$  gilt [vgl. 8, S. 12f.].

### Bemerkung 2

Ein Blockcode  $C$  ist  $\epsilon$ -fehlerkorrigierend, wenn  $d(C) \geq 2 \cdot \epsilon + 1$  gilt [vgl. 8, S. 12f.].

### Definition 8

Sei  $c \in C$  Codewort eines Blockcodes  $C \in A^n$  und  $B_t(c) = \{x \in A^n \mid d(x, c) \leq t\}$  die zu

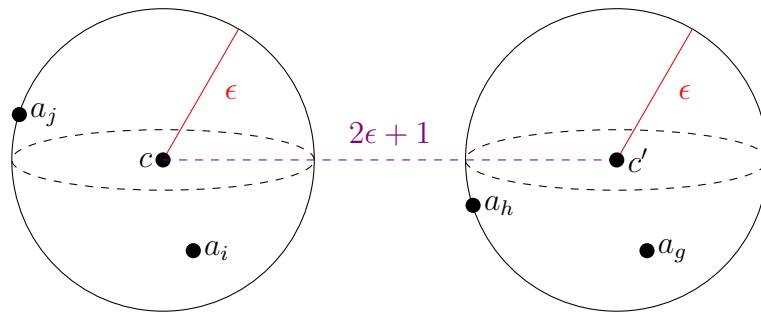


Abbildung 2.3: Visualisierung der Kugelinterpretation (nach [9])

$c$  gehörige Kugel mit allen Elementen aus  $A^n$ , die höchstens  $t$  entfernt sind.

Dann ist  $C$   **$t$ -fehlererkennend**, wenn  $\forall c \in C : B_t(c) \cap (C \setminus \{c\}) = \emptyset$ , die Kugel um  $c$  also keine anderen Codeworte enthält [vgl. 8, S. 13].

### Bemerkung 3

Ein Blockcode  $C$  ist  $t$ -fehlererkennend, wenn  $d(C) \geq t + 1$  gilt [vgl. 8, S. 13].

### Beispiel 3

Sei  $A = \{0, 1\}$  und  $A^3 = \{\langle 0, 0, 0 \rangle, \langle 0, 0, 1 \rangle, \dots, \langle 1, 1, 1 \rangle\}$ . Ferner sei  $C \subseteq A^3$  ein Blockcode für Nachrichten der Form  $m = \langle m_1, m_2 \rangle$  mit der Codierfunktion  $f: M \rightarrow C, \langle m_1, m_2 \rangle \mapsto \langle m_1, m_2, ((m_1 + m_2) \bmod 2) \rangle$ .

Für die Nachricht  $x = \langle 0, 1 \rangle$  ergibt sich also  $f(x) = \langle 0, 1, 1 \rangle$ .

Dann ist  $C = \{\langle 0, 0, 0 \rangle, \langle 0, 1, 1 \rangle, \langle 1, 0, 1 \rangle, \langle 1, 1, 0 \rangle\}$ . Daraus folgt  $d(C) = 2$ . Damit ist  $C$  1-fehlererkennend, da  $2 = 1 + 1$ , aber nicht fehlerkorrigierend, da  $2 \not\geq 2 \cdot 1 + 1$  gilt.

Welches Potenzial von fehlerkorrigierenden Codes für das zugrundeliegende Problem ausgeht, verdeutlicht das folgende Theorem:

## 2.3.2 Noisy Channels Coding Theorem

SHANNON konnte 1948 zeigen, dass es für einen binären, symmetrischen und rauschenden Kanal (BSC) Codes gibt, bei der die auftretende Fehlerwahrscheinlichkeit beliebig klein wird, solange sich die Datenrate der Übertragung unterhalb der Kapazität eines Kanals befindet.

**Theorem 1**

Sei  $C$  die **Kapazität** eines binären, rauschenden Kanals,  $n$  die Länge uniformer Codeworte,  $P(E)$  die Fehlerwahrscheinlichkeit und  $D$  die Datenübertragungsrate, wobei  $D < C$ . Dann gilt

$$P(E) \leq 2^{-n \cdot e(D)}$$

wobei  $e(D)$  eine vollständig von Kanalparametern abhängige positive Funktion, genannt **Fehlerexponent**, ist.

Folglich ist es möglich, Datenübertragungsvorgänge unter der Wahl einer geeigneten Datenübertragungsrate und einer Codierung nahezu fehlerfrei umzusetzen, völlig unabhängig von der Länge der zu übertragenden Nachricht. Es folgt auch, dass sich die Fehleranzahl nicht linear zur Fehleranfälligkeit (angegeben durch die Kapazität) des Kanals verhält [vgl. 4, S. 209ff.].

## 2.4 Finitfeldarithmetik

Viele der in der Praxis häufig eingesetzten Codierfunktionen basieren auf der Arithmetik in endlichen Körpern und *Galois*-Feldern, die im Folgenden eingeführt werden.

Ausgangspunkt der Definition von endlichen Körpern ist nach LIDL UND NIEDERREITER die algebraische Struktur der **Gruppe**.

### 2.4.1 Gruppen

**Definition 9**

Sei  $G$  eine Menge und  $\circ$  eine binäre Relation  $\circ : G \times G \rightarrow G$  mit folgenden Eigenschaften:

1. **Assoziativität:**

$$\forall a, b, c \in G : a \circ (b \circ c) \Leftrightarrow (a \circ b) \circ c$$

2. **Existenz eines neutralen Elements:**

$$\exists e \in G, \forall a \in G : a \circ e \Leftrightarrow e \circ a \Leftrightarrow a$$

3. **Existenz inverser Elemente:**

$$\forall a \in G, \exists a^{-1} \in G : a \circ a^{-1} \Leftrightarrow a^{-1} \circ a \Leftrightarrow e$$



Dann ist  $(G, \circ)$  eine **Gruppe**.

Gilt für eine Gruppe zudem  $\forall a, b \in G : a \circ b \Leftrightarrow b \circ a$ , so ist sie **kommutativ**, auch bezeichnet als **abelsch** [vgl. 12, S. 2].

#### Beispiel 4

Gegeben seien die Menge der ganzen Zahlen  $\mathbb{Z}$  und eine binäre Relation.

- $(\mathbb{Z}, +)$ , wobei  $+$  die Addition bezeichne, ist eine **abelsche Gruppe**, da die folgenden Eigenschaften erfüllt sind:

1. Assoziativität:

Seien  $a, b, c \in \mathbb{Z}$ . Dann gilt mit dem Distributivgesetz der Addition  $a + (b + c) \Leftrightarrow a + b + c \Leftrightarrow (a + b) + c$

2. Existenz eines neutralen Elements:

$$\forall x \in \mathbb{Z} : x + 0 \Leftrightarrow 0 + x \Leftrightarrow x$$

3. Existenz inverser Elemente:

$$\forall x \in \mathbb{Z}, \exists -x \in \mathbb{Z} : x + (-x) = 0$$

4. Kommutativität:

$$\forall a, b \in \mathbb{Z} : a + b \Leftrightarrow b + a$$

- $(\mathbb{Z}, \cdot)$ , wobei  $\cdot$  die Multiplikation bezeichne, ist keine Gruppe, da die inversen Elemente zu Elementen von  $\mathbb{Z}$  nicht Teil von  $\mathbb{Z}$  sind.

$$\forall x \in \mathbb{Z} \setminus \{1, -1\} : x \cdot \frac{1}{x} = 1, \text{ aber } \frac{1}{x} \notin \mathbb{Z}$$

#### Definition 10

Eine multiplikative Gruppe ist **zyklisch**, sofern

$$\exists a \in G, \forall b \in G, \exists j \in \mathbb{Z} : b = a^j$$

Dann ist  $a$  ein **Generator** der multiplikativen Gruppe<sup>2</sup>  $G$  [vgl. 12, S. 3f.]. Die Gruppe ist durch  $G = \langle a \rangle$  hinreichend beschrieben.

---

<sup>2</sup>Wird zu einer Gruppe keine binäre Relation angegeben, handelt es sich implizit um eine multiplikative Verknüpfung.

**Kryptographische Bedeutung** Zu Gruppen existieren einige mathematische Probleme, die sich für kryptographische Verfahren anbieten. MYASNIKOV, USHAKOV UND SHPILRAIN stellen in [13] kryptographische Protokolle dar, die beispielsweise auf dem *Konjugationssuchproblem* nicht-kommutativer Gruppen basieren:

Für zwei Elemente  $u, v \in G$  einer Gruppe  $G$  soll mindestens ein  $x \in G$  gefunden werden, sodass  $u^x = v$  gilt. Ist kein anderer Algorithmus für dieses Problem für die Gruppe  $G$  bekannt, so lässt sich  $x \rightarrow u^x$  als Einwegfunktion auffassen und so ein kryptographisches Protokoll konstruieren [vgl. 13, S. 37f.].

Abhängig davon, ob eine Gruppe abelsch ist oder nicht, lassen sich unterschiedliche Such- oder Entscheidungsprobleme zur Konstruktion kryptographischer Verfahren verwenden, die in [13] dargestellt, aber in dieser Arbeit nicht weiter behandelt werden.

## 2.4.2 Körper

### Definition 11

Ein **Ring**  $(R, +, \circ)$  ist eine nicht-leere Menge  $R$  mit den zwei auf ihr definierten binären Relationen, notiert durch  $+$  und  $\circ$ , für die gilt:

- $R$  ist abelsche Gruppe bezüglich der Operation  $+$
- Die Operation  $\circ$  ist assoziativ, folglich gilt

$$\forall a, b, c \in R : \quad (a \circ b) \circ c \Leftrightarrow a \circ (b \circ c)$$

- Das Distributivgesetz gilt, also ist

$$\forall a, b, c \in R : \quad (a \circ (b + c) \Leftrightarrow a \circ b + a \circ c) \wedge ((b + c) \circ a \Leftrightarrow b \circ a + c \circ a)$$

Ein Ring ist **kommutativ**, wenn die Operation  $\circ$  kommutativ ist. Ferner ist ein Ring ein **Divisionsring**, falls  $(R \setminus \{0\}, \circ)$  eine Gruppe bildet [vgl. 12, S. 13][vgl. 14, S. 1f.].

Ein Ring beinhaltet folglich eine Gruppe, jedoch auch eine Operation, die nicht notwendigerweise die Anforderungen an eine Gruppe erfüllt. Da beispielsweise der Menge  $\mathbb{Z}$  inverse Elemente bezüglich der Multiplikation fehlen, so ist  $(\mathbb{Z}, +, \circ)$  dennoch ein Ring, wenn auch kein Divisionsring.

**Definition 12**

Ein kommutativer Divisionsring wird als **Feld** oder **Körper** bezeichnet [vgl. 12, S. 13].

Um nun endliche Körper und *Galois-Körper* definieren zu können, ist das Konzept der **Restklasse** unabdingbar, das nun zunächst wieder auf Ringen definiert wird.

**Definition 13**

Sei  $R$  ein Ring. Dann ist  $J$  ein **Unterring** von  $R$ , falls  $J$  abgeschlossen ist unter beiden Operationen  $+$  und  $\circ$  [vgl. 12, S. 13].

**Bemerkung 4**

Abgeschlossenheit bezeichnet hier, dass das Ergebnis der Operation auf zwei beliebigen Elementen des Rings ein Element ergibt, das wiederum Element des Rings ist.

**Definition 14**

Ferner ist  $J$  ein **Ideal** von  $R$ , falls  $J$  ein Unterring von  $R$  ist und

$$\forall a \in J, r \in R : ((ar \in J) \wedge (ra \in J))$$

gilt [vgl. 12, S. 13].

Gilt  $I = (a)$  für ein  $a \in I$ , so ist  $I$  ein von  $a$  erzeugtes **Hauptideal**.

**Bemerkung 5**

Ein Ideal ist nun nicht mehr nur bezüglich eigenen Elementen und beiden Operationen abgeschlossen, sondern in Bezug auf die Multiplikation auch bezüglich Elementen des übergeordneten Rings.

**Definition 15**

Sei  $R$  eine Restklasse,  $a \in R$  und  $J$  ein Ideal von  $R$ . Die **Restklasse** von  $a \bmod J$  sei definiert als Menge aller Elemente der Form  $a + c \in R, \forall c \in J$  [vgl. 12, S. 13][vgl. 14, S. 10]:

$$[a] = a + J = \{a + c \in R : c \in J\}$$

**Bemerkung 6**

Zwei Elemente  $a, b \in R$  sind *kongruent* modulo  $J$ , wenn sie Elemente der gleichen Restklasse bezüglich  $J$  sind, geschrieben  $a \equiv b \bmod J$ .

Auch auf Restklassen lassen sich die Operationen eines Rings anwenden:

$$(a + J) + (b + J) = (a + b) + J$$

$$(a + J) \circ (b + J) = ab + J$$

### Bemerkung 7

Durch diese beiden Operationen auf Restklassen entsteht ein Ring, der als **Restklassenring** bezeichnet und mit  $R/J$  notiert wird [vgl. 12, S. 13f.] [vgl. 14, S. 10].

### Beispiel 5

Ein klassisches Beispiel für einen auf diese Weise entstandenen Restklassenring ist  $\mathbb{Z}/(p)$ ,  $p \in \mathbb{P}$ , wobei  $\mathbb{P}$  die Menge aller Primzahlen bezeichnet. Hier ist  $(p)$  ein durch  $p$  generiertes Hauptideal. Dieser Ring ist ein Körper, wie in [12] bewiesen wird und aufgrund der endlichen Anzahl an Elementen **finit**. Die Elemente sind beschrieben durch

$$\{[0] = 0 + (p), [1] = 1 + (p), \dots\}$$

Der Restklassenring  $\mathbb{Z}/(3)$  enthält die Elemente  $[0]$ ,  $[1]$  und  $[2]$ .

Nun ist die Definition der *Galois-Körper* unmittelbar möglich:

### Definition 16

Sei  $p \in \mathbb{P}$ . Dann sei  $\mathbb{F}_p$  der Körper, der durch den Restklassenring  $\mathbb{Z}/(p) = \mathbb{Z}/p\mathbb{Z}$  beschrieben wird und folglich die Struktur  $\mathbb{F}_p = \{[0], [1], \dots, [p-1]\}$  aufweist.  $\mathbb{F}_p$  wird dann als **Galois-Körper** der Ordnung  $p$  bezeichnet [vgl. 12, S. 15][vgl. 15, S. 75].

### Beispiel 6

Beispiele für endliche Körper sind:

- $\mathbb{F}_2 = \{[0], [1]\}$
- $\mathbb{F}_3 = \{[0], [1], [2]\}$ , beispielsweise  $\{0, 1, -1\}$

Neben *Galois-Körpern*, die auf diese Weise erzeugt wurden, bilden auch alle Primzahlpotenzen  $p^f$ ,  $p \in \mathbb{P}$  und  $f \in \mathbb{N}$  jeweils einen endlichen Körper.

**Theorem 2**

Für jede Primzahlpotenz  $p^f$ ,  $p \in \mathbb{P}$  und  $f \in \mathbb{N}$  gibt es (bis auf Isomorphie) genau einen endlichen Körper  $K$  mit  $|K| = p^f$ .

Die durch Primzahlpotenzen entstandenen endlichen Körper basieren nun nicht bloß auf den Restklassen des Unterkörpers  $\mathbb{F}_p$ , sondern werden mit **Polynomen** erweitert.

**2.4.3 Polynome**

Durch die Verkettung der beiden Operationen eines Körpers  $\mathbb{K}$  lassen sich **Polynome** erzeugen. Die Menge aller Polynome über einem Körper  $\mathbb{K}$  wird definiert als

$$\mathbb{K}[x] = \{p(x) = a_n x^n + \dots + a_1 x + a_0 \mid a_i \in \mathbb{K}\}$$

[vgl. 16, S. 118f.] und bildet einen kommutativen Ring, aufgrund der fehlenden Abgeschlossenheit der Division jedoch keinen Körper. Wird ein Polynom durch ein anderes dividiert, so kann dabei ein Rest entstehen, was die Anwendung des Kongruenzbegriffs nahelegt. Die Polynomdivision aller Polynome aus einem Körper  $\mathbb{K}$  mit einem Polynom  $m(x)$  erzeugt einen **Restklassenring**  $\mathbb{K}[x]/(m(x))$  mit der bekannten Addition und Multiplikation auf Polynomen.

Die Frage, ob jener Ring auch ein (endlicher) Körper ist, ergibt sich daraus, ob zu jedem Polynom ein multiplikatives Inverses Teil des Rings ist. Dafür ist folgende Eigenschaft eines Polynoms relevant:

**Definition 17**

Ein Polynom  $p(x) \in \mathbb{K}$  vom Grad (also dem höchsten Exponenten des Polynoms)  $\deg(p) > 1$  wird als **irreduzibel** bezeichnet, falls es kein Polynom  $q(x) \in \mathbb{K}$  mit  $0 < \deg(q) < \deg(p)$  gibt, das  $p(x)$  teilt [vgl. 16, S. 129].

Wird ein Polynomring nun durch reduzibles Polynom dividiert, also durch ein Polynom, dass sich selbst weiter dividieren lässt, existiert zu jedem seiner Teiler kein multiplikatives Inverses mehr, weshalb dadurch folglich kein Körper beschrieben wird.

**Theorem 3**

Ein Ring  $\mathbb{K}[x]/(m(x))$  ist genau dann ein Körper, wenn  $m(x)$  irreduzibel über  $\mathbb{K}$  ist [vgl. 16, S. 130].

Ein Galois-Körper  $\mathbb{F}_p$  kann auf diese Weise zu einem Körper  $\mathbb{F}_{p^k}$  erweitert werden, in dem aus dem Polynomring  $\mathbb{F}_p[x]$  ein irreduzibles Polynom  $m(x)$  vom Grad  $k$  ausgewählt wird. Dann ist

$$\mathbb{F}_{p^k} \quad \Leftrightarrow \quad \mathbb{F}_p[x]/(m(x))$$

**Beispiel 7**

Typische Beispiele für endliche Körper sind:

- Über dem Körper  $\mathbb{F}_2$  ist  $x^2+x+1$  ein irreduzibles Polynom, das zur Körpererweiterung genutzt werden kann und nun den endlichen Körper  $\mathbb{F}_{2^2} = \{[0], [1], [x], [x+1]\}$  erzeugt.
- Für das AES-Kryptosystem und eine Vielzahl weiterer kryptographischer und codierungstheoretischer Anwendungen wird der Körper  $\mathbb{F}_{2^8}$  mit dem definierenden Polynom  $x^8 + x^4 + x^3 + x + 1$  verwendet. Er enthält 256 Elemente [vgl. 16, S. 131].

### 3 Lineare fehlerkorrigierende Codes für kryptographische Zwecke

Mithilfe der vorgestellten Polynome soll nun eine Klasse von Blockcodes betrachtet werden, zu der auch *Reed-Solomon-Codes* und *Goppa-Codes*, die beiden Codes aus den Verfahren von NIEDERREITER und McELIECE, zählen. Diese Klasse wird als **lineare Codes** bezeichnet.

Für die Definition linearer Codes ist eine Wiederholung der algebraischen Struktur des **Vektorraums** notwendig. In [15] ist folgende Definition zu finden:

#### Definition 18

Sei  $K$  ein Körper. Eine Menge  $V$  mit einer additiven Verknüpfung

$$+: V \times V \rightarrow V; \quad \langle v, w \rangle \mapsto v + w$$

und einer Skalarmultiplikation

$$\circ: K \times V \rightarrow V; \quad \langle \lambda, v \rangle \mapsto \lambda \circ v$$

heißt  **$K$ -Vektorraum**, sofern gilt:

- $(V, +)$  ist **abelsche Gruppe**.
- Die Skalarmultiplikation ist sowohl bezüglich des Skalars als auch der Komponente distributiv und assoziativ, ferner ist das Element  $1 \in K$  das neutrale Element für diese Skalarmultiplikation. So muss  $\forall \lambda, \mu \in K; v, w \in V$  gelten:

$$(\lambda + \mu) \circ v = \lambda \circ v + \mu \circ v$$

$$\lambda \circ (v + w) = \lambda \circ v + \lambda \circ w$$

$$\lambda \circ (\mu \circ v) = (\lambda \circ \mu) \circ v$$

$$1 \circ v = v$$

[vgl. 15, S. 95]

Vektorräume können mit einer beliebigen Dimension konstruiert werden und beinhalten folglich Vektoren dieser Dimension, deren Komponenten addiert und mit Skalaren multipliziert werden können. Für die Definition linearer Codes ist es erforderlich, Vektorräume  $\mathbb{F}_q^n$  auf Primzahlkörpern  $\mathbb{F}_q$  zu definieren.

#### Theorem 4

Endliche Körper  $\mathbb{F}_q$  bilden  $\mathbb{F}_q^n$ -Vektorräume.

*Beweis.* Es bietet sich an, induktiv vorzugehen:

- **Annahme:**  $\mathbb{F}_q^n$  bildet einen Vektorraum  $V(q, n)$ .
- **Induktionsanfang:** Zu zeigen ist, dass  $\mathbb{F}_q^1$  die Forderungen eines Vektorraums erfüllt.
  - Die Forderung danach, dass  $(\mathbb{F}_q, +)$  eine abelsche Gruppe bildet, folgt unmittelbar aus der Körperdefinition.
  - Die Forderungen nach Distributivität und Assoziativität der Skalarmultiplikation bzw. Addition folgen direkt aus der Ringeigenschaft, da  $\lambda, \mu, v \in \mathbb{F}_q^1$ .
  - Die Existenz eines neutralen Elements bezüglich der Skalarmultiplikation ergibt sich daraus, dass gemäß Definition  $q \geq 2 > 1 \Rightarrow 1 \in \mathbb{F}_q$  gilt.
- **Induktionsschritt**  $1 \rightarrow n$ : Sei die komponentenweise Addition induktiv definiert als

$$(v_1, v_2, \dots, v_n) + (w_1, w_2, \dots, w_n) = (v_1 + w_1, v_2 + w_2, \dots, v_n + w_n)$$

und die Skalarmultiplikation als

$$k \circ (v_1, v_2, \dots, v_n) = (k \circ v_1, k \circ v_2, \dots, k \circ v_n)$$

. Alle obigen Forderungen folgen damit induktiv und  $\mathbb{F}_q^n$  ist ein  $\mathbb{F}_q$ -Vektorraum.

□

#### Definition 19

Ein **Untervektorraum**  $W$  eines Vektorraums  $V$  ist eine Teilmenge  $W \subset V$ , für die gilt:

- $W \neq \emptyset$
- $v, w \in W : \quad v + w \in W$



- $v \in W, \lambda \in K : \quad \lambda \circ v \in W$

[vgl. 15, S. 96]

### Definition 20

Sei  $\mathbb{F}_q$  ein endlicher Körper,  $|K| = q$ . Dann ist ein Blockcode  $C$  ein **linearer Code** mit dem Alphabet  $\mathbb{F}_q$  und Länge  $n$ , falls die Menge  $C$  ein **Untervektorraum** von  $\mathbb{F}_{q^n}$  ist [vgl. 8, S. 29].

Folglich beschreibt ein linearer Code Tupel der Länge  $n$  mit Komponenten aus  $\mathbb{F}_q$ . Ferner stellt die Untervektorraum-Eigenschaft sicher, dass die Addition von Codeworten und die Multiplikation eines Codewortes mit einem Skalar aus dem übergeordneten endlichen Körper  $\mathbb{F}_q$  weiterhin valide Codeworte erzeugt. Dies ist eine elementare Eigenschaft linearer Codes.

Eine Subklasse der linearen Codes sind die **zyklischen Codes**.

## 3.1 Zyklische Codes

### Definition 21

Ein  $k$ -dimensionaler Untervektorraum  $C$  von  $\mathbb{F}_q^n$  ist dann ein **zyklischer Code**, wenn

$$\forall \langle a_0, a_1, \dots, a_{n-1} \rangle \in C : \quad \langle a_{n-1}, a_0, a_1, \dots, a_{n-2} \rangle \in C$$

gilt [vgl. 5, S. 42].

Ein linearer Code ist folglich dann zyklisch, falls zu jedem Codewort auch das Codewort Teil des Codes ist, das durch eine zyklische Verschiebung entsteht. Ein solcher Code ist beispielsweise der *Reed-Colomon-Code*, der Gegenstand des nächsten Abschnitts ist.

### Theorem 5

Zyklische Codes der Länge  $n$  sind Ideale des Rings  $R = \mathbb{F}_q[x]/(x^n - 1)$ .

Das Polynom  $x^n - 1$  fungiert hierbei als Modul, sodass der Ring  $R$  folglich die Restklassen bezüglich dieser modularen Reduktion von  $\mathbb{F}_q$  enthält.

**Bemerkung 8**

Sei  $C$  ein linearer zyklischer Code der Länge  $n$  über einem Körper  $F_q$  und  $a = a(x) = a_0 + a_1 \cdot x + \dots + a_{n-1} \cdot x^{n-1} \in C$ . Dann ist

$$\begin{aligned} x \cdot a &= x \cdot (a_0 + a_1 \cdot x + \dots + a_{n-1} \cdot x^{n-1}) \\ &= a_0 \cdot x + a_1 \cdot x^2 + \dots + a_{n-1} \cdot x^n \\ &= a_{n-1} + a_0 \cdot x + \dots + a_{n-2} \cdot x^{n-1} \end{aligned}$$

genau das zyklisch verschobene Codewort  $a' \in C$ .

Dies folgt aus  $x^n \equiv 1 \pmod{x^n - 1}$  [vgl. 6, S. 149f.].

Da durch die Linearität des Codes  $C$  lässt sich ein  $a \in C$  nicht nur beliebig oft zyklisch verschieben, also mit  $x$  multiplizieren, sondern auch lineare Kombinationen (also Summen) gültiger Codeworte ergeben wiederum ein gültiges Codewort in  $C$ . Somit ergibt auch das Produkt eines Codewortes mit einem Polynom, also

$$a(x) \cdot c(x) \in C, c(x) \in C, a \in \mathbb{F}_q[x]/(x^n - 1)$$

ein gültiges Codewort [vgl. 17, S. 121].

Da Ideale eines Polynomrings erzeugende Polynome besitzen, ist die Angabe eines **Generatorpolynoms**  $g(x)$  möglich, für das gilt:

- $\deg g(x) = n - k$
- $g(x) \mid (x^n - 1)$

**Theorem 6**

Sei  $C$  ein linearer zyklischer Code über einem endlichen Körper  $\mathbb{F}_q$ . Dann gibt es ein Polynom  $g(x)$ , für das  $C = \langle g(x) \rangle$  gilt.

Ein Codewort beziehungsweise ein Codewortpolynom  $c(x)$  eines linearen, zyklischen Codes mit Generatorpolynom  $g(x)$  kann nun als Produkt von Nachrichtpolynom  $m(x)$  und Generatorpolynom aufgefasst werden [vgl. 17, S. 121].

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

Bei vielen aktuellen Anwendungen von *Reed-Solomon-Codes* werden Generatorpolynome angegeben und genutzt [vgl. 18, S. 6].

## 3.2 Definition der Klasse von Reed-Solomon-Codes

Die grundlegende Idee von linearen zyklischen Codes über Polynomen besteht darin, Fehler über Interpolation zu korrigieren. Bildlich gesehen soll also ein Codewort konstruiert werden, dessen einzelnen Komponenten entlang einer interpolierbaren Abbildung aufgereiht sind. Sollte nun ein Übertragungsfehler die Lage eines Punktes (also einer Komponente) verändern, so ist die korrekte Lage durch die anderen Punkte jedoch hinreichend bestimmt und kann korrigiert werden.

**Idee** Eine Nachricht der Länge  $k$  in Form eines Elementes eines endlichen Körpers  $\mathbb{F}_q^k$  kann als Polynom aufgefasst werden, indem die entsprechenden Komponenten der Nachricht die Koeffizienten des Polynoms bilden. Werden nun  $n$  paarweise verschiedene Stützstellen aus dem dem Code zugrundeliegenden Körper  $\mathbb{F}_q^n$  ausgewählt, so ergibt die Auswertung des Polynoms an den jeweiligen Stützstellen die Komponenten eines betrachteten Codewortes.

Durch bereits  $k$  Komponenten des Codewortes ist das Polynom hinreichend bestimmt, sodass  $n - k$  Punkte, die während der Übertragung ausgelöscht wurden, und  $(n - k)/2$  verfläuscht übertragene Punkte als Fehler erkannt und korrigiert werden können.

### Definition 22

Sei  $u = \langle u_1, u_2, \dots, u_n \rangle \in \mathbb{F}_q^n$ , wobei  $\forall i, j \in \{1, \dots, n\} : u_i \neq u_j$ . Dann ist  $a = \langle a_1, a_2, \dots, a_n \rangle \in \mathbb{F}_q^n$  ein Codewort eines **Reed-Solomon-Codes**  $C$  mit Länge  $n$ , Dimension  $k$  und Minimaldistanz  $d$ , falls es ein Polynom  $f(x)$  mit Grad  $\deg f < k$  gibt, sodass  $f(u_i) = a_i$  ist. Die Menge aller dieser Codeworte  $a$  ist  $C$ :

$$C = \{a \mid a_i = f(u_i), \deg f(x) < k, f \in \mathbb{F}_q[x]\}$$

[vgl. 19, S. 9]

**Beispiel 8**

Sei  $m = \langle 0, 1, 2, 1 \rangle \in \mathbb{F}_7^4$ . Dann ist  $m(X) = 0 + 1 \circ X + 2 \circ X^2 + 1 \circ X^3 \in \mathbb{F}_7[X]$  die polynomielle Darstellung dieser Nachricht  $m$ .

Betrachtet werde nun ein **Reed-Solomon-Code**  $C$  mit Länge  $n = 7$  und Dimension  $k = 4$ :

$$C = \{a \mid a_i = f(u_i), \deg f(x) < 4, f \in \mathbb{F}_7^7[x]\}$$

- Gewählt sind folglich  $n$  paarweise verschiedene Stützstellen, deren Wahlfreiheit in  $\mathbb{F}_7$  lediglich in der Anordnung besteht. Möglich ist beispielsweise  $u = \langle 0, 1, 2, 3, 4, 5, 6 \rangle \in \mathbb{F}_7^7$ .

Stützstelle $u_i$	0	1	2	3	4	5	6
Auswertung $m(u_i)$	0	4	4	6	2	5	0

- Folglich ergibt sich das Codewort  $c = \langle 0, 4, 4, 6, 2, 5, 0 \rangle \in \mathbb{F}_7^7$ .

**Bemerkung 9**

Das Generatorpolynom zu einem *Reed-Solomon-Code* ist definiert durch

$$g(x) = \prod_{j=1}^{2t} (x - \alpha^j)$$

wobei  $t$  die Anzahl der durch diesen Code korrigierbaren Fehler und  $\alpha$  ein primitives Element des zugrundeliegenden Körpers repräsentiert, es gilt  $n - k = 2t$  [vgl. 20, S. 17].

**3.2.1 Verallgemeinerte Reed-Solomon-Codes**

*Reed-Solomon-Codes* lassen sich verallgemeinern, in dem ein zusätzlicher Vektor  $v = \langle v_1, \dots, v_n \rangle$  eingebracht wird.

**Definition 23**

Sei  $u = \langle u_1, u_2, \dots, u_n \rangle \in \mathbb{F}_q^n$ , wobei  $\forall i, j \in \{1, \dots, n\}: u_i \neq u_j$ . Ferner sei  $v = \langle v_1, v_2, \dots, v_n \rangle \in \mathbb{F}_q^n \setminus \{0\}$ . Der **verallgemeinerte Reed-Solomon-Code**  $GRS_k(u, v)$  besteht nun aus der Menge der Codeworte

$$c = \langle v_1 f(u_1), v_2 f(u_2), \dots, v_n f(u_n) \rangle$$

für Polynome  $f \in \mathbb{F}_q[x]/(x^n - 1)$  mit Grad  $\deg f < k$  [vgl. 19, S. 13].

Verallgemeinerte Reed-Solomon-Codes sind in der Regel nicht zyklisch, folglich entfällt die Möglichkeit, ein Generatorpolynom anzugeben. Die Codierung basiert entsprechend auch nicht mehr auf einer simplen Multiplikation mit dem Generatorpolynom, sondern mit einer Generatormatrix. Verallgemeinerte Reed-Solomon-Codes finden Verwendung im Ansatz von Niederreiter und werden daher auch im Folgenden wieder betrachtet.

Nun wird in der Literatur gelegentlich auch der zu einem *Reed-Solomon-Code* **duale Code** verwendet:

### 3.2.2 Duale Codes

#### Definition 24

Sei  $C$  ein linearer  $[n, k, d]_q$ -Code mit Länge  $n$ , Dimension  $k$  und Minimaldistanz  $d$  über einem Körper  $\mathbb{F}_q$ . Ferner bezeichne die Operation  $\cdot$  das Skalarprodukt über dem Untervektorraum  $\mathbb{F}_q^n$ , das definiert ist durch

$$\langle a_1, a_2, \dots, a_n \rangle \cdot \langle b_1, b_2, \dots, b_n \rangle = \sum_{i=1}^n a_i b_i$$

Nun ist der zu  $C$  **duale Code**  $C^\perp$  definiert als

$$C^\perp = \{v \in \mathbb{F}_q^n \mid v \cdot c = 0, \forall c \in C\}$$

[vgl. 8, S. 35f.]

Diese Definition mag an die aus der linearen Algebra im Reellen bekannte Definition von Orthogonalität („Rechtwinkligkeit“) erinnern, jedoch lässt sich dies auf die hier betrachteten Körper nicht übertragen, unter Anderem, da es auch **selbstduale** Codes gibt [vgl. 8, S. 36].

#### Bemerkung 10

Der zu einem  $[n, k, d]_q$ -Code  $C$  duale Code  $C^\perp$  ist ein  $[n, n - k, d]_q$ -Code [vgl. 8, S. 36].

Wird der duale Code zu einem dualen Code eines linearen Codes  $C$  gebildet, so ist dieser wieder  $C$ , also gilt:

$$C^{\perp\perp} = C$$

**Theorem 7**

Der duale Code zu einem Reed-Solomon-Code ist ebenfalls ein Reed-Solomon-Code und der duale Code zu einem verallgemeinerten Reed-Solomon-Code ist ebenfalls ein verallgemeinerter Reed-Solomon-Code [vgl. 19, S. 14].

So gilt  $GRS_k(u, v)^\perp = GRS_{n-k}(u, v')$ , wobei für die Komponenten des Vektors  $v'$  gilt:

$$v_i \cdot v'_i \prod_{j=1, j \neq i}^n (u_j - u_i) = 1$$

[vgl. 19, S. 14]

**3.2.3 Kontroll- und Generatormatrizen**

Da lineare Codes wie gezeigt Vektorräume bilden, ist die Angabe von **Basen** möglich. Durch die Definition der Dualität von Codes zueinander, können **Generator-** und **Kontrollmatrizen** angegeben werden.

**Definition 25**

Sei  $C$  ein linearer  $[n, k, d]_q$ -Code und  $C^\perp$  der duale Code zu  $C$ .

Ferner sei  $(\langle h_{11}, h_{12}, \dots, h_{1n} \rangle, \langle h_{21}, \dots, h_{2n} \rangle, \dots, \langle h_{(n-k)1}, \dots, h_{(n-k)n} \rangle)$  eine Basis zu  $C^\perp$ . Dann ist die Matrix

$$H = \begin{pmatrix} h_{11} & h_{12} & \dots & h_{1n} \\ h_{21} & h_{22} & \dots & h_{2n} \\ \dots & \dots & \dots & \dots \\ h_{(n-k)1} & h_{(n-k)2} & \dots & h_{(n-k)n} \end{pmatrix}$$

die **Generatormatrix** von  $C^\perp$  und **Kontrollmatrix** oder **Paritätsprüfmatrix** von  $C$  [vgl. 8, S. 37]

Der Zusammenhang zwischen Generator- und Kontrollmatrix ergibt sich daraus, dass über die Generatormatrix des dualen Codes  $C^\perp$  zu  $C$  überprüft werden kann, ob für ein  $x \in \mathbb{F}_q$  auch  $x \in C$  gilt, indem die Skalarprodukte  $x \cdot h_i, \forall i \in \{1, \dots, n-k\}$  gebildet werden. Sollte für mindestens eines dieser Produkte  $x \cdot h_i \neq 0$  gelten, so ist  $x \notin C$  [vgl. 8, S. 37]. So ließe sich beispielsweise eine Identitätsabbildung  $1_C(x)$  auf  $C$  bilden.

### Generatormatrizen des verallgemeinerten Reed-Solomon-Codes

Zur Kodierung eines Informationstupels mittels des verallgemeinerten Reed-Solomon-Codes wird die Generatormatrix des Codes benötigt.

#### Bemerkung 11

Gegeben sei  $GRS_k(u, v)$  und der dazu duale Code  $GRS_{n-k}(u, y)$ . Die Paritätsprüfmatrix zu  $GRS_k(u, v)$  ist dann genau die Generatormatrix zu  $GRS_{n-k}(u, y)$  und definiert durch

$$H = (y_i u_i^j)_{0 \leq j \leq n-k-1, 1 \leq i \leq n} = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \\ y_1 u_1 & y_2 u_2 & \cdots & y_n u_n \\ \vdots & \vdots & & \vdots \\ y_1 u_1^{n-k-1} & y_2 u_2^{n-k-1} & \cdots & y_n u_n^{n-k-1} \end{pmatrix}$$

Entsprechend ist die Generatormatrix zu  $GRS_k(u, v)$  definiert durch

$$G = (v_i u_i^j)_{0 \leq j \leq k-1, 1 \leq i \leq n} = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \\ v_1 u_1 & v_2 u_2 & \cdots & v_n u_n \\ \vdots & \vdots & & \vdots \\ v_1 u_1^{k-1} & v_2 u_2^{k-1} & \cdots & v_n u_n^{k-1} \end{pmatrix}$$

[vgl. 17, S. 277][vgl. 21, S. 304].

### 3.3 QR-Codes als Anwendungsbeispiel

*Reed-Solomon-Codes* finden aufgrund ihrer vergleichsweise einfachen Codierbarkeit Anwendung in einer Vielzahl von Problemstellungen des Alltags: Neben der Raumfahrt oder der korrekten Wiedergabe von CDs sind sie auch Teil von *QR-Codes*, die im Folgenden als Anwendungsbeispiel betrachtet werden.

### 3.3.1 Hintergrund und Konzept

*Quick Response (QR)-Codes*<sup>1</sup> sind zweidimensionale Matrix-Codes, die 1994 von einer Tochterfirma des Automobilherstellers Toyota entwickelt wurden, ursprünglich, um die Nachverfolgung von Fahrzeugkomponenten zu gewährleisten [vgl. 22, S. 1]. Ziel der Entwicklung war es, ein Barcode-Format zu entwickeln, dass hochperformant und einfach zu scannen ist [vgl. 22, S. 1]. Kodierbare Daten sind beispielsweise alphanumerische Zeichenketten, URLs oder Bytedata.

#### Fehlerkorrektur

Zwei Arten von Fehlern sind möglich:

- **Erase** – Ein Symbol, das nicht gescannt oder dekodiert werden kann
- **Error** – Ein Symbol, das falsch dekodiert wurde

Durch die Matrixform des Codes führt ein irrtümlich als hell oder dunkel erkanntes Symbol weiterhin zu einem gültigen, wenn auch falschen Codewort. Zur Korrektur dieses falschen Codewortes sind zwei Korrekturworte erforderlich [vgl. 23, S. 33].

Je nach gewählter Qualitätsstufe ist eine verschiedene Anzahl von Korrekturworten erforderlich, die den Tabellen der ISO-Norm 18004 in der jeweils gültigen Fassung entnommen werden können [vgl. 23, S. 35-44]. Zu jedem Datenwort (bei Bedarf mit Padding-Worten) wird ein Fehlerkorrekturwort berechnet und an die Daten angefügt.

### 3.3.2 Konstruktion von Fehlerkorrekturworten

Verwendet wird der Galois-Körper  $\mathbb{F}_{2^8}$  mit dem definierenden und auf diesem Körper irreduziblen Polynom  $x^8 + x^4 + x^3 + x^2 + 1$ .

Die **Datencodeworte** bilden nun die Koeffizienten eines Polynoms, wobei das erste Datencodewort die Koeffizienten des Terms mit dem höchsten Grad repräsentieren und das letzte Datencodewort entsprechend die Koeffizienten des Terms mit dem niedrigsten Grad [vgl. 23, S. 45].

---

<sup>1</sup>Aufgrund des fehlenden Zugriffs auf die spezifizierende ISO-Norm in der aktuellen Fassung bezieht sich diese Arbeit auf den Stand von 2000.



Dieses Polynom wird nun durch ein für jede QR-Code-Konfiguration in [23] definiertes **Generatorpolynom** dividiert. Der dabei entstehende Rest bildet nun die **Fehlerkorrekturworte**, indem der Koeffizient des Terms mit dem höchsten Grad das erste Fehlerkorrekturwort und der Koeffizient des Terms mit Grad 0 das letzte Fehlerkorrekturwort definiert [vgl. 23, S. 45].

Die nun generierten Datencodeworte und Fehlerkorrekturworte ergeben nach einer spezifizierten Anordnung nun die grafische Repräsentation des QR-Codes:

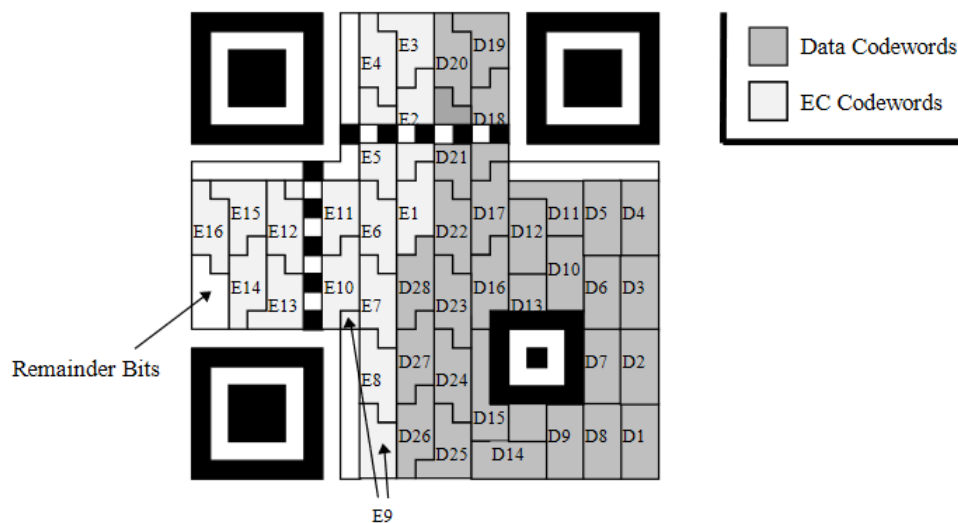


Figure 15 — Symbol character arrangement in version 2-M symbol

Abbildung 3.1: Anordnung einzelner Datencodeworte und Fehlerkorrekturcodeworte in einem QR-Code [23, S. 48]

### 3.3.3 Dekodierung

Für die folgenden Schritte wird ein *QR-Code* in der Version 1-M betrachtet. Gemäß der Norm wird für diese Version der  $[26, 16, 4]_{28}$ -Reed-Solomon-Code verwendet.

1. **Demaskierung** – Die Daten des QR-Codes sind zum Zweck einer besseren Scan-Qualität maskiert worden, diese Maskierung wird als erstes entfernt, es ergibt sich

das empfangene Codewort (mit potenziellen Fehlern)

$$R(x) = r_0 + r_1x^1 + \dots + r_{25}x^{25}$$

wobei  $\forall i \in \{1, \dots, 25\} : r_i \in \mathbb{F}_{2^8}$ .

2. **Syndrombildung** – Ausgehend von der Annahme, dass die Daten als Summe  $c(x) + e(x)$  von Codewortpolynom und einem Fehlerpolynom vorliegen, soll nun bestimmt werden, ob Fehler vorliegen. Dafür werden alle **Syndrome**  $S_i = R(\alpha^j)$ ,  $\forall j \in \{1, \dots, 2t\} \wedge i \in \{0, \dots, 8\}$  wobei  $\alpha$  primitives Element des Körpers  $\mathbb{F}_q$  ist, gebildet, da das Codewortpolynom für alle Wurzeln des Generatorpolynoms (also die Nullstellen) gemäß dem Nullproduktsatz 0 ergibt und somit lediglich die Fehlerpolynome verbleiben.
3. **Fehlerpositionsermittlung** – Für die Ermittlung der Fehlerpositionen wird folgendes Gleichungssystem aus den Syndromen gebildet:

$$S_0Y_4 - S_1Y_3 + S_2Y_2 - S_3Y_1 + S_4 = 0$$

$$S_1Y_4 - S_2Y_3 + S_3Y_2 - S_4Y_1 + S_5 = 0$$

$$S_2Y_4 - S_3Y_3 + S_4Y_2 - S_5Y_1 + S_6 = 0$$

$$S_3Y_4 - S_4Y_3 + S_5Y_2 - S_6Y_1 + S_7 = 0$$

Da Addition und Subtraktion in einem endlichen Körper der Primcharakteristik 2 gleichbedeutend sind, führen die alternierenden Vorzeichen nicht zu einer Abweichung vom später dargestellten Allgemeinfall.

Dieses Gleichungssystem hat mindestens eine nicht-triviale Lösung, sofern die tatsächliche Fehleranzahl  $w(e) \leq t$  ist. Mit dieser Lösung ergibt sich das **Fehlerlokalisationspolynom**

$$\sigma(x) = Y_4 + Y_3x + Y_2x^2 + Y_1x^3 + x^4$$

4. Falls  $\sigma(\alpha^j) = 0$ ,  $j \in \{0, \dots, n-1\}$ , so liegt an dieser Stelle  $j$  ein Fehler vor. Die Menge aller Fehlerstellen  $N(\sigma)$  sei definiert als  $\{i = \{1, \dots, n\} \mid \sigma(\alpha^i) = 0\}$ .
5. **Fehlerwertbestimmung** – Nun werden die einzelnen Fehlerwerte durch folgendes

Gleichungssystem berechnet:

$$\begin{aligned}
 E_{i_1} \cdot \alpha^{1j_1} + E_{i_2} \cdot \alpha^{1j_2} + \dots + E_{i_{w(e)}} \cdot \alpha^{1j_{w(e)}} &= S_0 \\
 E_{i_1} \cdot \alpha^{2j_1} + E_{i_2} \cdot \alpha^{2j_2} + \dots + E_{i_{w(e)}} \cdot \alpha^{2j_{w(e)}} &= S_1 \\
 &\dots \\
 E_{i_1} \cdot \alpha^{(n-k-1)j_1} + E_{i_2} \cdot \alpha^{(n-k-1)j_2} + \dots + E_{i_{w(e)}} \cdot \alpha^{(n-k-1)j_{w(e)}} &= S_{(n-k-1)}
 \end{aligned}$$

6. **Fehlerkorrektur** – Sollte das obige Gleichungssystem eine nicht-triviale Lösung haben, so lassen sich die Fehler korrigieren. Sei  $(\epsilon_{i_1}, \epsilon_{i_2}, \dots, \epsilon_{i_{w(e)}})$  diese Lösung. Sie wird nun zu einem alle Stellen umfassenden Tupel  $(\epsilon_1, \dots, \epsilon_n)$  erweitert, wobei für alle Stellen  $i \notin N(\sigma)$ :  $\epsilon_i = 0$  gilt. Die eigentliche Korrektur wird nun über

$$C = R - (\epsilon_1, \dots, \epsilon_n)$$

ausgeführt, wobei  $C$  nun das korrigierte Codewort darstellt.

[vgl. 23, S. 74f.]

## 3.4 Algorithmische Betrachtung

Ziel dieses Abschnittes ist es, die Kodierung und Dekodierung von *Reed-Solomon-Codes* als implementierbare Algorithmen zu zeigen, um anschließend darauf aufbauend ein kryptographisches Verfahren entwickeln zu können.

### 3.4.1 Kodierung von Reed-Solomon-Codes

Eine Nachricht  $m$  soll ergänzt um ein Fehlerkorrekturcodewort  $c$  über einen Kanal übertragen werden. Sender und Empfänger haben sich vor Datenübermittlung auf die Parameter  $n, k, d$  und  $q$  eines  $[n, k, d]_q$ -*Reed-Solomon-Codes* geeinigt.

1. Bilden des **Generatorpolynoms** mit  $\alpha$  als primitivem Element

$$g(x) = (x - \alpha^1) \cdot (x - \alpha^2) \cdot (x - \alpha^3) \cdot \dots \cdot (x - \alpha^{2t})$$

2. Bildung des Polynoms der Nachricht ( $m \rightarrow m(x)$ )

$$m(x) = m_0 + m_1x + m_2x^2 + \dots + m_{k-1}x^{k-1}, \forall i \in \{0, \dots, k-1\}: m_i \in \mathbb{F}_q$$

3. Berechnen des **Fehlerkorrekturwortes**

$$r(x) = m(x) \cdot x^{n-k} \mod g(x)$$

Es ist bekannt, dass  $c(x) = m(x)g(x)$  gilt. Diese Vorgehensweise hat jedoch den Nachteil, dass sich die zur Fehlerkorrektur zusätzlich eingebrachten Prüfbits an Stellen des Polynoms befinden, die im Codewort eine saubere Trennung in Informations- und Prüfbits unmöglich machen. Daher wird ein anderer Ansatz gewählt, bei dem die zusätzlichen Prüfbits an die Nachricht angehängt werden, während die Nachricht selbst unverändert übernommen wird [vgl. 8, S. 127f.]

4. **Zusammensetzen**

$$c(x) = m(x) \cdot x^{n-k} + r(x)$$

Würde hier bloß  $c(x) = m(x) + r(x)$  angegeben sein, so könnte nicht ausgeschlossen werden, dass sich Terme gleichen Grades gegenseitig aufheben. Daher wird der Informationsteil des Codewortes mittels Multiplikation um  $x^{n-k}$  an höhere Potenzen verschoben. In einer technischen Realisierung werden zu diesem Zweck Schieberegister eingesetzt [vgl. 8, S. 126f.]. Es ergibt sich folglich ein Codewort

$$c = \langle r_0, \dots, r_{n-k-1}, m_{n-k}, m_{n-k+1}, \dots, m_{n-1} \rangle$$

[vgl. 20, S. 17]

**Implementierung**

Das folgende Listing zeigt den systematischen Kodieralgorithmus nach [20] und [8] in einer Implementierung im Mathematiksoftwaresystem *SageMath* [24].

```

1  #!/usr/bin/env sage
2  # ...
3
4  # Basic Code properties
5  q = input("Please type in q, e.g. 2^2:\t")

```

```

6  n = int(input("Please type in length n: \t"))
7  k = int(input("Please type in dimension k: \t"))
8
9  # Generate Galois field
10 F_q = None
11 if len(q.split("^")) < 2:
12     F_q.<a> = GF(int(q))
13 elif len(q.split("^")) == 2:
14     F_q.<a> = GF(pow(int(q.split("^")[0]), int(q.split("^")[1])))
15 else:
16     print(f"Sorry, unable to work with this q={q}!")
17
18 # Generate RS-Code
19 R.<x> = PolynomialRing(F_q)
20 g = prod(x - a**i for i in range(1, n-k+1))
21 print(f"\nGenerator Polynomial: \t\t{g}")
22
23 # Convert Message m=(m_0, m_1, ..., m_{k-1}) to polynomial
24 m = input(f"\nPlease type in message tuple\n
25 \t\t(m_0,m_1,...,m_{k-1}) of F_{q}^{k}: \t")
26 m = [F_q(x) for x in m[1:-1].split(",")]
27
28 if len(m) != k:
29     print("\nERROR: The provided message has not exactly k digits!")
30     exit()
31
32 m_poly = 0
33 i = 0
34 while i < len(m):
35     m_poly += (m[i])*(x**i)
36     i += 1
37
38 print(f"\nMessage Polynomial: \t\t{m_poly}")
39
40 # Coding steps
41 (_, r) = (m_poly * x**(n-k)).quo_rem(g)

```

```

42 print(f"\nRemainder_Polynomial: \t\t{r}")
43
44 m_shifted = (m_poly * x**(n-k))
45 print(f"\nShifted_Message_Polynomial: \t\t{m_shifted}")
46
47 c = m_shifted + r
48 print(f"Concatenated_Codeword: \t\t{c}")
49
50 coefficients = c.coefficients(sparse=False)
51
52 while len(coefficients) < n:
53     coefficients = [0]+coefficients
54
55 print(f"\nCodeword_Coefficients: \t\t{coefficients}")

```

Quelltext 3.1: SageMath-Implementierung des Kodieralgorithmus

### 3.4.2 Fehlerkorrektur und Dekodierung

Das im Vergleich zur Kodierung schwierigere Problem, auf dem später auch die kryptographische Sicherheit aufgebaut wird, ist das der Dekodierung und der darin enthaltenen Fehlerkorrektur. Ziel dieses Abschnittes ist es, den syndrombasierten Dekodieralgorithmus nach Peterson, Gorenstein und Zierler darzustellen. Performantere Algorithmen, die an praktischer Bedeutung gewinnen, werden erwähnt, jedoch nicht im Detail dargestellt.

**Definitionen** Im Folgenden werden die folgenden Definitionen implizit verwendet:

- $R$  bezeichne die möglicherweise **verfälschte Nachricht**, die nach einer Datenübertragung über einen unsicheren Kanal empfangen wurde.
- $C$  bezeichne das dazugehörige fehlerfreie bzw. korrigierte **Codewort**
- $E$  bezeichne das **Fehlerwort**, das ebenfalls die Länge  $n$  von  $R$  und  $C$  hat, aber nur in  $e$  Komponenten ungleich 0 ist
- **Fehlerkoordinaten** bezeichnen die Stellen von  $R$ , an denen eine Abweichung im Vergleich zu  $C$  feststellbar ist

- Die **Fehlstellennummer** ist definiert als  $\alpha^j$ , sofern an der  $j$ -ten Komponente ein Fehler aufgetreten ist und  $\alpha$  primitives Element des betrachteten endlichen Körpers  $\mathbb{F}_q$  ist. Eine solche Fehlstellennummer wird mit  $a_k$  bezeichnet, wobei  $\{a_1, \dots, a_e\}$  folglich alle Fehlstellen identifiziert.
- Die **Fehlergröße** ist definiert als das Element aus  $\mathbb{F}_q$ , dass in der Fehlerkoordinate  $j$  auftritt. Zu einer Fehlstelle  $a_k$  bezeichnet  $b_k$  deren Fehlergröße.  $\{b_1, \dots, b_e\}$  identifiziert folglich alle Fehlergrößen.

[vgl. 25, S. 147f.]

$$R = C + E \Leftrightarrow (r_0, r_1, \dots, r_n) = (c_0, c_1, \dots, c_n) + (e_0, e_1, \dots, e_n)$$

$$r(x) = c(x) + e(x) \Leftrightarrow r_0 + r_1x + \dots + r_nx^n = c_0 + c_1x + \dots + c_nx^n + e_0 + e_1x + \dots + e_nx^n$$

Sei  $\alpha$  ein primitives Element des Körpers  $\mathbb{F}_q$ , wobei  $q = p^m, m \geq 1 \in \mathbb{N}$  gilt. Ferner sei  $g(x)$  das Generatorpolynom eines  $[n, k, d]_q$ -Reed-Solomon-Codes mit Fehlerkorrekturkapazität  $t = \lfloor \frac{n-k}{2} \rfloor$ . Dann gilt:

$$\forall i \in \{0, \dots, 2t\}: g(\alpha^i) = 0$$

Da  $c(x) = m(x)g(x)$  gilt damit sofort  $r(\alpha^i) = e(\alpha^i)$ . Dies ist der Einstieg in den Fehlerkorrekturalgorithmus:

1. Bilden der **Syndrome**  $S_j$  für jede Wurzel  $\alpha^j$  von  $g(x)$ :

$$S_j = r(\alpha^j) = e(\alpha^j) = \sum_{i=1}^t b_i a_i^j$$

Die Summe ergibt sich aus der Polynomdarstellung des Fehlerwortes, wobei  $\forall a^j = \alpha^{h \cdot j}$  für ein  $h$ . Alle  $a_i$  und  $b_i$  sind zu diesem Zeitpunkt jedoch unbekannt und durch das Potenzieren von  $a_i$  mit  $j$  auch nicht mehr linear. Dadurch wird das Hauptproblem des Dekodierens beschrieben [vgl. 25, S. 148].

2. Zum weiteren Vorgehen werde ein **Fehlerlokalisationspolynom**  $\sigma(x)$  konstruiert, dessen Wurzeln alle  $a_i \in \{a_1, \dots, a_e\}$  sind:

$$\sigma(x) = (x - a_1)(x - a_2) \dots (x - a_e)$$

und für den in der Praxis häufigen Fall  $p = 2$  auch

$$\sigma(x) = (a_1 + x)(a_2 + x) \dots (a_e + x)$$



# Literaturverzeichnis

- [1] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C*, 20th anniversary edition. Indianapolis: John Wiley und Sons, 2015, ISBN: 978-1-119-09672-6. Adresse: <https://www.schneier.com/books/applied-cryptography-2preface/> (besucht am 06.11.2022).
- [2] D. J. Bernstein, J. Buchmann und E. Dahmen, Hrsg., *Post-Quantum Cryptography*. Heidelberg: Springer, 2009, ISBN: 978-3-540-88701-0.
- [3] V. Cerf, „ASCII format for Network Interchange,“ RFC Editor, RFC 20, Okt. 1969, S. 1–56. DOI: 10.17487/RFC0020.
- [4] M. Borda, *Fundamentals in Information Theory and Coding*, 1. Aufl. Berlin: Springer, 2011. DOI: 10.1007/978-3-642-20347-3.
- [5] J. H. van Lint, *Coding Theory*, Lecture Notes in Mathematics, A. Dold und B. Eckmann, Hrsg. Berlin: Springer, 1973, ISBN: 3-540-06363-3.
- [6] N. L. Biggs, *Codes: An Introduction to Information Communication and Cryptography* (Springer Undergraduate Mathematics Series). London: Springer, 2008, DHBW-Bestand. DOI: 10.1007/978-1-84800-273-9.
- [7] W. Willems, *Codierungstheorie und Kryptographie*. Basel: Birkhäuser, 2008. DOI: 10.1007/978-3-7643-8612-2.
- [8] O. Manz, *Fehlerkorrigierende Codes*. Wiesbaden: Springer Vieweg, 2017. DOI: <https://doi.org/10.1007/978-3-658-14652-8>.
- [9] R. W. Hamming, „Error detecting and error correcting codes,“ *The Bell system technical journal*, Jg. 29, Nr. 2, S. 147–160, 1950. DOI: 10.1002/j.1538-7305.1950.tb00463.x.
- [10] A. Viterbi, „Convolutional Codes and Their Performance in Communication Systems,“ *IEEE Transactions on Communication Technology*, Jg. 19, Nr. 5, S. 751–772, 1971. DOI: 10.1109/TCOM.1971.1090700.
- [11] S. Roman, *Coding and Information Theory* (Graduate Texts in Mathematics). New York: Springer, 1992, Bd. 134, ISBN: 3-540-97812-7.

- [12] R. Lidl und H. Niederreiter, „Algebraic Foundations,“ in *Finite Fields* (Encyclopedia of Mathematics and its Applications), 2. Aufl., Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1996, S. 1–46, UB-Bestand. DOI: 10.1017/CB09780511525926.003.
- [13] A. Myasnikov, A. Ushakov und V. Shpilrain, *Group-based Cryptography* (Advanced Courses in Mathematics - CRM Barcelona). Basel: Birkhäuser, 2008, DHBW-Bestand. DOI: 10.1007/978-3-7643-8827-0.
- [14] J. M. Howie, *Fields and Galois Theory* (Springer Undergraduate Mathematics Series). London: Springer, 2006, UB-Bestand. DOI: 10.1007/978-1-84628-181-5.
- [15] G. Fischer und B. Springborn, *Lineare Algebra, Eine Einführung für Studienanfänger* (Grundkurs Mathematik), 19. Aufl. Berlin: Springer, 2020. DOI: 10.1007/978-3-662-61645-1.
- [16] G. Teschl und S. Teschl, *Mathematik für Informatiker, Band 1: Diskrete Mathematik und Lineare Algebra* (eXamen.press), 4. Aufl. Heidelberg: Springer, 2013. DOI: 10.1007/978-3-642-37972-7.
- [17] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. New York: Wiley-Interscience, 2005, ISBN: 978-0-471-64800-0.
- [18] S. B. Wicker und V. K. Bhargava, „An Introduction to Reed-Solomon Codes,“ in *Reed-Solomon Codes and Their Applications*. 1994, S. 1–16. DOI: 10.1109/9780470546345.ch1.
- [19] C. Mahr, *Zur Codierung von RS-, BCH- und Goppa-Codes* (Fortschritt-Berichte 95). Düsseldorf: VDI-Verlag, 1988, Bd. 10, UB-Bestand (Magazin).
- [20] N. Zivic, *Coding and Cryptography, Synergy for a Robust Communication*. München: Oldenbourg Wissenschaftsverlag, 2013, DHBW-Bestand. DOI: 10.1524/9783486781267.
- [21] „10 Reed-Solomon and Justesen codes,“ in *The Theory of Error-Correcting Codes*, Ser. North-Holland Mathematical Library, F. MacWilliams und N. Sloane, Hrsg., Bd. 16, Elsevier, 1977, S. 294–316. DOI: [https://doi.org/10.1016/S0924-6509\(08\)70535-X](https://doi.org/10.1016/S0924-6509(08)70535-X). Adresse: <https://www.sciencedirect.com/science/article/pii/S092465090870535X>.
- [22] S. Tiwari, „An Introduction to QR Code Technology,“ in *2016 International Conference on Information Technology (ICIT)*, 2016, S. 39–44. DOI: 10.1109/ICIT.2016.021.

- [23] „Information technology – Automatic identification and data capture techniques – Bar code symbology – QR Code,“ International Organization for Standardization, Geneva, CH, Standard, 2000.
- [24] „SageMath - Open-Source Mathematical Software System,“ SageMath Open Source Collective. (2022), Adresse: <https://www.sagemath.org/> (besucht am 17. 01. 2023).
- [25] D. G. Hoffman, D. A. Leonard, C. C. Lindner, K. T. Phelps, C. Rodger und J. R. Wall, *Coding Theory: The Essentials*. New York: Marcel Dekker, Inc., 1991, KIT-Bestand, ISBN: 0-8247-8611-4.