Skript zur Vorlesung

# Signale und Codes

Willi Geiselmann

**Karlsruher Institut für Technologie**

Fakultät für Informatik

Institut für Theoretische Informatik
Arbeitsgruppe für Kryptographie und Sicherheit

Letzte Änderung: 17. März 2016

# Inhaltsverzeichnis

# 1 Einleitung

Wozu Codierung?
Bei der Übertragung einer Nachricht vom Sender zum Empfänger können Fehler auftreten, so dass die empfangene Nachricht nicht lesbar wird. Fehler treten auf durch Störungen im Kommunikationskanal, wie zum Beispiel Umgebungsrauschen. Auch bei der Speicherung von Daten können diese durch z. B. Entmagnetisierung der Festplatte verloren gehen.
Um dies zu verhindern bzw. vermindern, werden die Nachrichten so codiert, damit Fehler erkannt und korrigiert werden können. Ziel ist es also, Fehler bei der Übertragung und der Speicherung zu finden bzw. zu korrigieren.

## 1.1 Ein einfaches Beispiel

### 1.1.1 Hamming's Problem

Bis vor dem Jahr 1940 wurden Nachrichten durch Wiederholen codiert. Das heißt, jedes Bit wurde z. B. dreifach geschrieben. Per Mehrheitsentscheid wurde dann das "richtige" Bit ausgewertet. Dies ist ein relativ guter und einfacher Code, jedoch verbraucht diese Lösung den 3-fachen Speicherplatz. Zusätzlich darf natürlich nur 1 Fehler in jedem 3-Bit-Block auftauchen.
Im Jahr 1940 kam Richard Wesley Hamming zu einem anderen Lösungsansatz. Hamming hatte die Idee die Nachricht $m$ in 4-Bit-Blöcken (anstelle von einem Bit) aufzuteilen.
Ein Codewort $c$ berechnet sich dann aus dem Code $C = G \cdot m \pmod 2$ mit der Nachricht $m$ und einer Generatormatrix $G$.

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} \\ \left.\vphantom{\begin{matrix} a \\ a \\ a \\ a \end{matrix}}\right\} \end{matrix} \begin{matrix} \text{alle Spaltenvektoren mit Gewicht} \geq 2 \\ \\ \text{Einheitsmatrix } E_4 \end{matrix}$$

#### Eigenschaften des Hamming-Codes

Der Hamming-Code hat $2^4 = 16$ Codeworte der Länge 7. Im Gegensatz zur bisherigen Codierung hat eine 4 Bit Nachricht eine 3 Bit Redundanz. Das bedeutet, die Länge hat sich knapp verdoppelt, anstatt wie bisher verdreifacht.

#### Kann dieser Code auch einen 1-Bit Fehler korrigieren?

Dies gilt genau dann, wenn 2 Codeworte $c_1 = G \cdot m_1$ und $c_2 = G \cdot m_2$, wobei $m_1 \neq m_2$ ist, mindestens den Hamming-Abstand 3 haben. Das bedeutet, die zwei Codeworte unterscheiden sich mindestens an 3 Stellen. Durch einfaches Durchprobieren kann dies überprüft werden, dafür sind nur 16 Codeworte zu vergleichen.
Mit etwas "Theorie" geht es besser:
Die Menge der Codeworte bilden einen Vektorraum der Dimension 4 über $\mathbb{F}_2$, erzeugt von der Matrix $G$. Jeder Vektorraum kann als Kern einer Matrix beschrieben werden.

**Gesucht:**
Eine Prüfmatrix $H$ mit $H \cdot c = H \cdot G \cdot m = 0$ $\forall c \in C$ bzw. $\forall m \in \mathbb{F}_2^4$

**Vorschlag:**

$$H := \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Wie findet man nun den Kern von $H$? Dazu gibt es einen Trick in der Linearen Algebra:

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ & & & & & & 1 \end{pmatrix}$$

Da offensichtlich $G$ im Kern von $H$ liegt, gilt:

$$H \cdot G = 0 \textbf{ und } Kern(H) = C.$$

Tritt ein 1-Bitfehler auf, dann gilt für das Wort $d \notin C$:

$$H \cdot d \neq 0.$$

Es genügt also zu zeigen, dass zwei verschiedene Vektoren im $Kern(H)$ den Abstand $\geq 3$ haben.
Ist $d(c_1, c_2)$ der Abstand von $c_1$ und $c_2$, dann gilt:
Das Gewicht $wgt(c_1 + c_2) = d(c_1, c_2)$.
Das heißt, es genügt zu zeigen, dass es kein Codewort (außer 0) gibt mit Gewicht $< 3$.

Beispiel:
Die Vektoren $c_1 = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \end{pmatrix}^t$ und $c_2 = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \end{pmatrix}^t$ haben den Abstand 2, unterscheiden sich also an 2 Stellen. Das Gewicht von $c_1 + c_2 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \end{pmatrix}^t$ ist auch 2.

**Gesucht:**
Gesucht sind Vektoren aus $Kern(H)$ mit kleinem Gewicht bzw. existieren Vektoren mit Gewicht 1 oder 2?

- Vektoren mit Gewicht 1 liefert eine Spalte von H ($\neq 0$)
- Vektoren mit Gewicht 2 liefern Summe von 2 Spalten von H ($\neq 0$, da je 2 Spalten linear unabhängig sind)
- Gewicht 3 existiert (Spalten von $G$)

$\Rightarrow$ 1 Fehler kann korrigiert werden

Die Matrix $H$ hat die Eigenschaft, dass je zwei Spalten linear unabhängig sind. Es kann also keinen Vektor in $Kern(H)$ geben, welcher das Gewicht 1 oder 2 hat. Das heißt, alle Vektoren im $Kern(H)$ unterscheiden sich an mindestens 3 Stellen. Womit gezeigt ist, dass zwei Codeworte $c_1 = G \cdot b_1$ und $c_2 = G \cdot b_2$ sich an 3 Stellen unterscheiden, somit den Hamming-Abstand 3 besitzen.

### 1.1.2 Decodieren

Um den 1-Bit Fehler zu finden, kann nun die Kontrollmatrix $H$ verwendet werden:

Empfangen wurde also das Wort $c + e_i$ (mit $i \in \{1...7\}$), wobei $e_i$ der Fehlervektor ist, also ist das $i$-te Bit falsch. Dann liefert die Multiplikation mit $H$:

$$\begin{aligned} H \cdot (c + e_i) &= H \cdot c + H \cdot e_i \\ &= 0 + H \cdot e_i \end{aligned}$$

Das Ergebnis von $H(c + e_i)$ ist die $i$-te Spalte von $H$, also ist das $i$-te Bit falsch.

# 2 Block Codes

Ein Block Code codiert eine Nachricht in Worte fester Länge. Wichtige Parameter eines Blockcodes sind die Informationsrate (Kenngröße für die in einer festen Datenmenge enthaltenen Informationsmenge) und die Korrekturleistung (der Hamming-Abstand, eine Kenngröße für die Fehlerresistenz bei einer festen Datenmenge). Eine „Verbesserung“ des einen Parameters führt üblicherweise zu einer „Verschlechterung“ des anderen Parameters.

**Definition 1.** Es sei Q ein Alphabet und $q := \#Q$,

- dann heißt eine nichtleere Teilmenge $C \subseteq Q^n$ ein *Blockcode*;
- ist $\#C = 1$, besteht der Code also nur aus einem Wort, so heißt $C$ *trivial*;
- falls $q = 2$, so heißt $C$ *binärer Code*.

**Definition 2.** Seien $x, y \in Q^n$, dann heißt $d(x, y) := \#\{i \mid i = 1, \ldots, n, ; x_i \neq y_i\}$ der *Hammingabstand* von $x$ und $y$.

Der Hammingabstand ist ein Maß für Symbol-(Bit-)Fehler. Er ist ein Maß im Sinne der Maßtheorie, es gilt also $d(x, y) = d(y, x)$ und die Dreiecksungleichung:

**Lemma 1** (Dreiecksungleichung)**.** *Für alle $x, y, z \in Q^n$ gilt $d(x, z) \leq d(x, y) + d(y, z)$.*

**Beweis**. Es gilt:

$$\begin{aligned} d(x, z) &= \#\{i|x_i \neq z_i\} \leq \#\{i|x_i \neq y_i \vee y_i \neq z_i\} \\ &= \#\left(\{i|x_i \neq y_i\} \cup \{i|y_i \neq z_i\}\right) \\ &\leq \#\{i|x_i \neq y_i\} + \#\{i|y_i \neq z_i\} = d(x, y) + d(y, z) \end{aligned}$$

□

**Definition 3.** Sei $C$ ein Block Code, dann heißt

$$d(C) := \min\{d(c_1, c_2) \mid c_1, c_2 \in C; c_1 \neq c_2\}$$

der *Minimalabstand* von C.

Einer der naheliegendsten Begriffe für den sich die Codierungstheorie interessiert ist die Informationsrate, sie ist das Verhältnis der übertragenen Nachricht und der darin enthaltenen Information:

**Definition 4.** Es sei $C$ ein Block Code, dann heißt

$$R(C) := \frac{log_q(\#C)}{log_q(\#Q^n)} = \frac{log_q(\#C)}{n \cdot log_q(q)} = \frac{log_q(\#C)}{n}$$

die *Rate von C*.

Auf den ersten Blick kann die Definition über Logarithmen seltsam erscheinen, betrachtet man aber das folgende Beispiel, so sieht man schnell ein, dass dies die einzige sinnvolle Definition ist.

**Beispiel.** Betrachtete den $[7, 4]$-Hamming Code aus dem vorherigen Abschnitt. Er hat die folgenden Eckdaten:

*a)* $Q = \{0, 1\}$
$\#C = 2^4 = 16$
$\#Q^7 = 2^7 = 128$

*b)* Hängt man zwei Codeworte hintereinander, so ergibt sich für diesen Code $\overline{C}$:
$Q = \{0, 1\}$
$\#\overline{C} = 2^8 = 256$
$\#Q^{14} = 2^{14} = 2^{14}$

Würde man die Rate ohne den Logarithmus berechnen, so ergäbe dies eine Rate von $\frac{1}{8}$ für a) und $\frac{1}{64}$ für b). Bei der Definition mit Logarithmus sind die Raten der beiden Codes jeweils $\frac{4}{7}$ und damit gleich, wie man es erwarten sollte.

Der zweite zentrale Begriff der Codierungstheorie ist die Korrekturleistung eines Codes: Hat der Blockcode $C$ die Minimaldistanz $d$, wie viele Fehler können dann erkannt oder korrigiert werden?

**Lemma 2.** *Sei $C \subset Q^n$ ein Code mit Minimalabstand $d(C) = d$, dann können entweder*

*(1)* $d - 1$ *Fehler erkannt*

*oder*

*(2)* $\lfloor \frac{d-1}{2} \rfloor$ *Fehler korrigiert*

*werden.*

**Beweis**.

(1) Anschaulich: „$d-1$ Fehler bzw. Änderungen liefern kein anderes Codewort!“
Formal: Sei $c \in C$ und $\tilde{c} \in Q^n$ mit $c \neq \tilde{c}$ und $d(c, \tilde{c}) \leq d-1$. Dann ist $\tilde{c} \notin C$. Andernfalls wäre $d(C) \leq d-1$, was ein Widerspruch zu der Annahme $d(C) = d$ ist. Somit lassen sich $d-1$ Fehler bei der Übertragung erkennen.

(2) Sei $c \in C$ und $\tilde{c} \in Q^n$ mit $d(c, \tilde{c}) \leq \lfloor \frac{d-1}{2} \rfloor$.
Behauptung: $c$ ist das einzige Codewort mit $d(c, \tilde{c}) \leq \lfloor \frac{d-1}{2} \rfloor$.
Sonst existiert ein $\hat{c} \in C$ mit $d(\tilde{c}, \hat{c}) \leq \lfloor \frac{d-1}{2} \rfloor$ und dann folgt: $d(c, \hat{c}) \leq d(c, \tilde{c}) + d(\tilde{c}, \hat{c}) \leq 2 \cdot \lfloor \frac{d-1}{2} \rfloor \leq d-1$. Dies steht im Widerspruch zu $d(C) = d$, also lässt sich $\tilde{c}$ eindeutig zu $c$ decodieren.

□

# 3 Lineare Block-Codes

Ein linearer Block-Code ist ein spezieller Blockcode, bei dem die Codewörter Elemente eines endlichdimensionalen Vektorraums $\mathbb{F}_q^n$ über einem endlichen Körper $\mathbb{F}_q$ sind. Der Code ist genau dann ein linearer Blockcode, wenn er ein Untervektorraum von $\mathbb{F}_q^n$ ist. Der Vorteil zum allgemeine Code ist, dass Methoden der linearen Algebra verwendet werden können. Ist der lineare Code $C$ nun ein Untervektor von $\mathbb{F}_q^n$ der Dimension $k$, dann nennen wir $C$ einen $[n, k]$-linearen Blockcode, er hat die Informationsrate

$$R(C) = \frac{log(q^k)}{log(q^n)} = \frac{k \cdot \cancel{log(q)}}{n \cdot \cancel{log(q)}} = \frac{k}{n}.$$

**Einschub: Endliche Körper**

Genau für $q = p^r$, wobei $r > 0$ und $p$ eine Primzahl ist, existieren Körper $\mathbb{F}_q$ mit $q$ Elementen. Für $r = 1$ ist $\mathbb{F}_p \cong \mathbb{Z}_p$ ein Körper mit der gewohnten modulo Arithmetik in $\mathbb{Z}$. Für $r > 1$ ist $\mathbb{F}_q \neq \mathbb{Z}_q$, z. B. ist $\mathbb{Z}/8\mathbb{Z}$ kein Körper. Einige Eigenschaften und Beispiele von endlichen Körpern finden sich im Anhang A.

**Definition 5.** Für $x \in \mathbb{F}_q^n$ heißt $wgt(x) := d(x, 0)$ das *Hamming Gewicht* von $x$.

Für das Gewicht gelten die folgenden Rechenregeln:

$$d(x, y) = wgt(x - y).$$

$$wgt(x + y) = d(x, -y) \leq d(x, 0) + d(y, 0) = wgt(x) + wgt(y)$$

**Satz 3.** *Es sei $C$ ein $[n, k]$-Code über $\mathbb{F}_q$ mit $d(C) = d$. Dann existiert ein $c \in C$ mit $wgt(c) = d$. Das bedeutet, die Minimaldistanz $d(C)$ ist das kleinste Gewicht eines Codeworts $c \neq 0$. Daher nennt man $d(C)$ auch das* Minimalgewicht *von $C$.*

**Beweis**. Sei $c_1, c_2 \in C$ mit $d(c_1, c_2) = d$, dann gilt:

$$d(c_1, c_2) = d(c_1 - c_2, c_2 - c_2) = d(c_1 - c_2, 0) = wgt(c_1 - c_2)$$

wobei aus $c_1, c_2 \in C$ wegen der Abgeschlossenheit der Addition in einem Vektorraum auch $(c_1 - c_2) \in C$. □

Damit ist das Bestimmen von $d(C)$ bei linearen Codes deutlich einfacher als bei allgemeinen Codes. Es müssen nicht alle Paare von Codeworte betrachtet werden, es reichen die einzelnen Codeworte.
Lineare Codes lassen sich einfach durch Matrizen beschreiben: $C$ ist ein Vektorraum und lässt sich somit durch seine Basis beschreiben.

**Definition 6.** Für jeden $[n, k]$-Code $C$ über $\mathbb{F}_q$ existiert eine $(n \times k)$-Matrix $G$ vom Rang $k$ mit den Basisvektoren von $C$ in den Spalten der Matrix. Die Matrix $G$ heißt *Generatormatrix* von $C$.

Mithilfe der Generatormatrix $G$ können effizient Nachrichten $m \in \mathbb{F}_q^k$ auf Codeworte $c$ abgebildet werden durch $c := G \cdot m$.

**Definition 7.** Es sei $C$ ein $[n, k]$-Code und $G$ eine Generatormatrix von $C$, dann heißt eine $((n - k) \times n)$-Matrix $H$ vom Rang $(n - k)$ mit $H \cdot G = 0$ *Prüfmatrix* von $C$.

Der Vektorraum $C$ lässt sich auch als Kern einer Matrix $H$ darstellen, wobei $C$ ein $[n, k]$-Code ist. Dann gilt für eine Prüfmatrix $H$ von $C$: $C = kern(H) = \{c \in \mathbb{F}_q^n \mid H \cdot c = 0\}$.
Um zu prüfen, ob $\tilde{c} \in \mathbb{F}_q^n$ ein Codewort ist, kann man die folgende Äquivalenz benutzen:

$$\tilde{c} \in C \Leftrightarrow H \cdot \tilde{c} = 0.$$

Die Prüfmatrix ist ein wichtiges Hilfsmittel bei der Decodierung von Worten, man kann mit ihr aber auch die Minimaldistanz des Codes berechnen:

**Satz 4.** *Es sei $H$ eine Prüfmatrix eines $[n, k, d]$-Codes $C$. Dann ist $d$ gleich der minimalen Anzahl linear abhängiger Spalten von $H$.*

**Beweis**. Sei $c \in C$ mit $wgt(c) = d$. Dann gilt $H \cdot c = 0$. Das heißt:
Es gibt $d$ linear abhängige Spalten von $H$, die Spalten mit den Indizes an denen $c$ nicht Null ist.
Umgekehrt gilt für jedes $\tilde{c}$ mit $wgt(\tilde{c}) = d - 1$, dass $\tilde{c} \notin C$. Also gilt $H \cdot \tilde{c} \neq 0$ und daher sind je $d - 1$ Spalten von $H$ linear unabhängig. □

## 3.1 Syndromdecodierung

Können wir aus $H \cdot \tilde{c} = s \neq 0$ noch mehr Information erhalten als nur $\tilde{c} \notin C$?
Dazu definieren wir uns zunächst:

**Definition 8.** Es sei $C$ ein $[n, k]$-Code, $\tilde{c} \in \mathbb{F}_q^n$ und $H$ eine Prüfmatrix von $C$, dann heißt $s := H \cdot \tilde{c}$ das *Syndrom* von $\tilde{c}$.

Ist beim Senden eines Codeworts $c$ ein Fehler $e$ aufgetreten, dann gilt:

$$\begin{aligned} H(c+e) &= H \cdot c + H \cdot e \\ &= 0 + H \cdot e \end{aligned}$$

Das Syndrom hängt folglich nur vom Fehler $e$ ab und nicht vom Codewort $c$. Ist kein Fehler vorhanden, so ergibt das Syndrom den Nullvektor. Liegt ein Übertragungsfehler $e$ vor, so lässt sich über das Syndrom der Fehler $e$ eindeutig charakterisieren.
Wie bereits beschrieben, kann $C$ bis zu $\lfloor \frac{d-1}{2} \rfloor$ Fehler korrigieren. Um den Fehler $e$ zu finden und zu korrigieren, wird $H \cdot e$ für alle $e \in \mathbb{F}_q^n$ mit $wgt(e) \leq \lfloor \frac{d-1}{2} \rfloor$ berechnet. Dazu wird eine Tabelle erstellt mit den Spalten $(e, H \cdot e)$.
Wird nun ein Wort $\tilde{c}$ empfangen, so berechnet man das Syndrom $H \cdot \tilde{c}$ und suche es in der Tabelle. Das entsprechende $e$ ist dann der Fehler und $\tilde{c}$ wird zu $\tilde{c} - e$ decodiert.
Dieses Vorgehen muss nicht immer funktionieren: Hat ein Code z. B. das Minimalgewicht $d(C) = 4$ und ein empfangenes Wort hat Abstand 2 vom nächsten Codewort, so wird der Fehler mit Gewicht 2 nicht in der Tabelle erscheinen (es werden nur Fehler $\leq \frac{4-1}{2} = 1,5$ berücksichtigt). Es kann aber sein dass bei manchen Fehlern vom Gewicht 2 nur ein Codewort mit Abstand 2 existiert und damit eine eindeutige Decodierung möglich wäre. Wir können diese Decodierung recht einfach durchführen, indem wir unsere Syndromtabelle auch für größere Fehlergewicht berechnen, wir müssen dazu lediglich wissen wann wir aufhören können. Dies liefert uns die folgende einfache Überlegung:

- Es gibt $2^n$ mögliche Worte
- zu jedem Syndrom gehören $2^k$ Worte

$\Rightarrow$ es gibt $\frac{2^n}{2^k} = 2^{n-k}$ Syndrome.

Wir können beim Aufbau der Tabelle also nach $2^{n-k}$ gefundenen Fehlermustern aufhören zu suchen.

## 3.2 Systematische Codierung

Aus der linearen Algebra wissen wir, dass es zu einem Vektorraum normalerweise mehrere verschiedene Basen gibt und ein Basiswechsel zu verschiedenen, ähnlichen Matrizen (zur Darstellung von linearen Abbildungen) führt.
Vor diesem Hintergrund ist es naheliegend zu untersuchen ob es verschiedene Generator- bzw. Prüfmatrizen eines Codes $C$ gibt und ob sich darin eine „Normalform“ finden/definieren läßt.
Im Allgemeinen sind $G$ und $H$ nicht eindeutig. Sei $G$ die Generatormatrix eines $[n, k]$-Codes von $C$, dann ist $\widetilde{G} = G \cdot T$ auch eine Generatormatrix von $C$ für jede invertierbare $k \times k$-Matrix $T$. ($T \cdot m$ ist eine “Umcodierung“ von Nachrichten, die wieder alle Vektoren in $\mathbb{F}_q^k$ liefert.)
Analog gilt für die Prüfmatrix $H$ von $C$, dass $S \cdot H$ auch eine Prüfmatrix ist für jede invertierbare $(n-k) \times (n-k)$-Matrix $S$.

Eine für die Codierung recht naheliegende Normalfor für $G$ wäre

$$\overline{G} = \left(\begin{array}{ccc} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \\ \hline & P & \end{array}\right). \tag{1}$$

Hier wird die Nachricht als Anfang des Codeworts verwendet und dann entsprechende Prüfzeichen angehängt.
Diese Normalform ist eindeutig. Eine Transformation mit einer Matrix $T$ liefert:

$$\overline{G} \cdot T = \left(\frac{E_k}{P}\right) \cdot T = \left(\frac{E_k \cdot T}{P \cdot T}\right)$$

Wird jetzt wieder die gewünschte Normalform erreicht, dann muss $E_k \cdot T = E_k$ gelten und damit ist $T = E_k$.
Eine entsprechende Normalform von $H$ ist $(-P \mid E_{n-k}) =: \overline{H}$. Es gilt

$$\overline{H} \cdot \overline{G} = (-P \mid E_{n-k}) \left(\frac{E_k}{P}\right) = -P \cdot E_k + E_{n-k} \cdot P = -P + P = 0,$$

daher ist $\overline{H}$ Prüfmatrix. Vorteile dieser Normalform sind:

- das Nachrichtenwort ist direkt im Codewort enthalten,
- nur $P$ muss gespeichert werden.

Jetzt stellt sich die Frage ob eine solche Normalform für jeden Code existiert. Die einfache, leider nicht befriedigende Antwort liefert ein Gegenbeispiel: Sei $G$ die folgende Generatormatrix eines $[4, 3]$-Codes.

$$G := \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Diese Matrix lässt sich nicht auf die Normalform bringen. Durch beliebiges $T \in GL(3, \mathbb{F}_2)$ wird

$$\overline{G} = G \cdot T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot T = \left(\begin{array}{c} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} \cdot T \\ \hline \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \cdot T \end{array}\right)$$

Eine obere $3 \times 3$-Matrix mit Rang 2 und kann durch Multiplikation mit $T$ nicht zu einer Einheitsmatrix von Rang 3 transformiert werden.
Um die Normalform doch noch zu erreichen erlauben wir es den Code zu modifizieren, so dass die Normalform dann für den modifizierten Code existiert. Vertauscht man in $G$ einige Zeilen, so dass oben $k$ linear unabhängige Zeilen stehen, so ändert dies nur die Reihenfolge der Stellen des Codeworts. Gewicht, Abstand von Codewörtern usw. bleibt erhalten. Daher definieren wir:

**Definition 9.** Es sei $C_1$ ein $[n,k]$-Code und $G_1$ eine Generatormatrix von $C_1$, dann heißt ein $[n,k]$-Code $C_2$ mit Generatormatrix $G_2$ äquivalent zu $C_1$, falls eine Permutationsmatrix $S$ und eine invertierbare Matrix $T$ existieren, so dass $G_1 = S \cdot G_2 \cdot T$ gilt.

**Satz 5.** *Zu jedem linearen Code $C$ mit Generatormatrix $G$ existiert ein äquivalenter Code mit Generatormatirx $\overline{G}$, so dass $\overline{G}$ die systematische Form von Gleichung (1) hat.*

**Beweis**. Die Matrix $G$ hat Rank $k$, also existieren $k$ linear unabhängige Zeilen. Ordne die $n$ Zeilen von $G$ so um, dass die ersten $k$ Zeilen linear unabhängig sind (dies wird durch die Multiplikation mit einer Permutationsmatrix $S$ erreicht):

$$S \cdot G = \left(\frac{U}{V}\right),$$

wobei $U$ nach Konstruktion von $S$ eine invertierbare $k \times k$-Matrix ist. Mit der Transformationsmatrix $T := U^{-1}$ ergibt sich die Nomalform. □

## 3.3 Hamming Codes

Die Grundidee des Hamming Codes ist, dass der Code einen einzigen Fehler korrigieren kann, also die Minimaldistanz $d_{min} = 3$ hat. Zusätzlich soll der Code möglichst wenig Redundanz aufweisen.
Im vorangegangenen Abschnitt wurde bewiesen, dass die minimale Anzahl der linear abhängigen Spalten einer Prüfmatrix $H$ dem Minimalabstand entsprechen. Das heißt, dass die Prüfmatrix eines Hamming Codes keine 2 Spalten besitzt, die linear abhängig sind.
Für binäre Vektoren gilt, dass zwei Vektoren $\neq 0$ linear unabhängig sind wenn sie ungleich sind. Über $\mathbb{F}_q$ mit $q \neq 2$ sind zwei Vektoren ($\neq 0$) linear unabhängig, wenn sie nicht Vielfache von einander sind.
Wie findet man nun eine Prüfmatrix bzw. Generatormatrix mit diesen Eigenschaften? Hierzu wähle die Anzahl der Zeilen der Prüfmatrix als einen festen Wert $\ell (= n-k)$ bei unbekanntem $n, k$.

$$H = \left.\begin{pmatrix} \cdots \\ \cdots \\ \cdots \end{pmatrix}\right\} \text{Anzahl der Zeilen } \ell = (n-k)$$

Somit gehört $H$ zu einem $[n, \overbrace{n-\ell}^{k}, 3]$-Code. Bei möglichst großem $n$ wird die Redundanz entsprechend am geringsten da die Rate des Codes $R = \frac{n-\ell}{n}$ mit steigendem $n$ auch steigt. Mit diesen beiden Bedingungen (größte Datenrate und 1-Fehler soll korregierbar sein) findet sich die Prüfmatrix für binäre Codes also indem die Prüfmatrix alle $2^\ell - 1$ vielen Worte $\neq 0$ der Länge $\ell$ als Spalten enthält.

$$H := \underbrace{\begin{pmatrix} 0 & & & & 1 \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ 0 & & & & 1 \\ 1 & & & & 1 \end{pmatrix}}_{2^\ell - 1}$$

**Satz 6.** *Die eben konstruierte Prüfmatrix gehört zu dem binären Hamming Code mit den Parametern* $[2^\ell - 1, 2^\ell - 1 - \ell, 3]$.

Der nicht binärer Hamming Codes über $\mathbb{F}_q$ hat bei gegebenen $\ell$ eine Länge $n := \frac{q^\ell - 1}{q-1}$; er ist ein $[n, n-\ell, 3]$-Code.

## 3.4 Perfekte Codes

Betrachten wir die Decodierung für Worte $\tilde{c} \in \mathbb{F}_2^n$ mit Abstand kleiner oder gleich $\frac{d-1}{2}$ zum nächsten Codewort. Wieviele Worte sind dann über diese „Decodierkugeln“ decodierbar? Legen wir dazu um einen Punkt $y \in \mathbb{F}_2^n$ eine Kugel $B(y, r)$ mit dem Radius $r$:

$$B(y, r) := \{\overline{y} \in \mathbb{F}_2^n \mid d(\overline{y}, y) \leq r\}$$

Die Anzahl $\mathrm{Vol}(r, n)$ der Punkte in dieser Kugel berechnet sich zu

$$\mathrm{Vol}(r, n) = \underbrace{1}_{\text{Punkt selbst}} + \underbrace{\binom{n}{1}}_{\text{Gewicht 1}} + \underbrace{\binom{n}{2}}_{\text{Gewicht 2}} + \ldots + \binom{n}{r}$$

Legen wir jetzt um jedes Codewort eines $[n, k, d]$-Codes $C$ eine Kugel mit dem Radius $\lfloor \frac{d-1}{2} \rfloor$. Dann stellt sich die Frage, ob es Worte außerhalb dieser Kugel gibt, oder ob jedes empfangene Wort in einer Kugel vom Radius $\lfloor \frac{d-1}{2} \rfloor$ liegt und damit auch eindeutig decodierbar ist.
Die Kugeln um alle Codewörter füllen maximal den gesamten Raum aus, also:

**Satz 7.** *Es sei $C$ ein $[n, k, d]$-Code über $\mathbb{F}_2$. Dann gilt*

$$\underbrace{2^k}_{\textit{Anzahl Codewörter}} \cdot \mathit{Vol}\left(\lfloor \frac{d-1}{2} \rfloor, n\right) \leq 2^n.$$

*Diese Schranke heißt* Hamming Schranke.

**Definition 10.** Ein Code, der die Ungleichung der Hamming Schranke mit „=“ erfüllt, heißt *perfekt.*

**Satz 8.** *Hamming Codes sind perfekt.*

**Beweis**. Sei $C$ der binär Hamming Code für ein festel $\ell \geq 1$, also ein $[2^\ell - 1, 2^\ell - 1 - \ell, 3]$-Code. Dann gilt:

$$\begin{array}{rcl} 2^k \cdot \mathrm{Vol}(1, n) & \leq & 2^n \\ \Rightarrow \underbrace{2^{2^\ell - 1 - \ell}(2^\ell - 1 + 1)}_{2^{2^\ell - 1}} & \leq & 2^n \\ 2^n & = & 2^n \end{array}$$

und die Hamming Schranke wird mit Gleichheit erfüllt. Für nicht binäre Hamming Codes ergibt sich die Gleichheit mit einer ähnlichen Rechnung. □

Es existieren nicht viele weitere perfekte Codes. Binäre perfekte Codes gibt es nur noch den Golay Code $G_{23}$ (vgl. Abschnitt 6.1) und die Wiederholungscodes mit ungerader Länge. Die folgenden Beispiele veranschaulichen die „Perfektheit“ der Wiederholungscodes:

1. Dreifache Wiederholung:

   $a \mapsto (a, a, a)$ mit $a \in \mathbb{F}_2$

   Es gibt nur zwei Codewörter und zwar $(0, 0, 0)$ und $(1, 1, 1)$. Der Radius der Kugel um ein Codewort ist $\lfloor \frac{d-1}{2} \rfloor = 1$. Die Hamming Schranke ergibt:

   $$2 \cdot \underbrace{(1 + 3)}_{\text{Vol}(1,3)} = 2^3$$

2. Wiederholung ungerader Länge $n$.

   $a \mapsto (a, \ldots, a)$ mit $a \in \mathbb{F}_2$

   Es gibt 2 Codewörter $(0, \ldots, 0)$ und $(1, \ldots, 1)$. Der Radius der Kugel ist $\lfloor \frac{d-1}{2} \rfloor = \lfloor \frac{n-1}{2} \rfloor$. Die Hamming Schranke ergibt:

   $$2 \cdot \underbrace{\sum_{r=0}^{\frac{n-1}{2}} \binom{n}{r}}_{\text{Vol}(1, \frac{n-1}{2})} = 2^n$$

3. Dreifache Wiederholung bei einem nicht binärer Code ($q := 3$).

   Es gibt die Codewörter $(0, 0, 0)$, $(1, 1, 1)$ und $(2, 2, 2)$.

   Dann hat $(1, 2, 0)$ zu jedem Codewort mit Abstand 2 und liegt damit in keiner der 3 Kugeln mit Radius 1. Es gibt Worte außerhalb der Decodierkugeln vom Radius $\frac{d-1}{2}$, der Code kann **nicht** perfekt sein.

**Satz 9.** *Es sei $C$ ein $[n, k, 2m + 1]$-Code, also mit ungerader Minimalanstand, ergänze die Generatormatrix um ein „Paritätsbit“ auf gerade Parität.*
*Der resultierende Code ist ein $[n + 1, k, 2m + 2]$-Code.*

**Beweis**. Alle Codeworte $\neq 0$ haben gerade Parität, also ein gerades Gewicht $\geq 2m + 1$. □

# 4 Schranken für Codes

## 4.1 Untere Schranke von unstrukturierten Codes

Betrachte einen strukturlosen Code $C$ über dem Alphabet $\Sigma$ (mit $|\Sigma| = q$), also $C \subseteq \Sigma^n$ mit Minimalabstand $d$.
Wollen wir einen Code konstruieren, der diesen Minimalabstand hat, so können wir z.B. folgendermaßen vorgehen: Lege eine Liste $L$ mit Wörtern aus $\Sigma^n$ an. Initialisiere den Code $C$ als leere Menge.
Nun wird aus $L$ ein Wort $x$ zufällig ausgewählt und zu $C$ hinzugefügt. Lösche nun alle Worte aus der Liste $L$, die einen Minimalabstand kleiner als $d$ zu dem gewählten Wort $x$ haben. Wiederhole diesen Vorgang bis die Liste $L$ leer ist. Der Algorithmus sieht dann wie folgt aus:

- Setze $L := \Sigma^n$, $C := \emptyset$
- Wiederhole:
  - Wähle $x \in L$ zufällig
  - $C := C \cup x$
  - Lösche in $L$ alle Worte $\overline{x}$ mit $d(x, \overline{x}) \leq d - 1$

  bis $L = \emptyset$ ist.

Dieses Vorgehen liefert bereits eine einfache Schranke für die Anzahl von Codeworten von $C$. Bei jedem neuen Codewort werden maximal $Vol_q(n, d-1)$ Elemente aus $L$ gestrichen, also gilt:

$$\begin{array}{rrcl} & |C| & \geq & \frac{q^n}{Vol_q(n,d-1)} \\ \Rightarrow & \log_q(|C|) & \geq & n - \log_q(Vol_q(n, d-1)) \end{array}$$

**Lemma 10.** *(Covering bound) Für $n, k, d, q$, mit $k \leq n - log_q(Vol_q(n, d-1))$ existiert ein (unstrukturierter) Code $C$ mit $q^k = |C|$ vielen Codewörtern. In Anlehnung an lineare Codes nennen wir $C$ einen $(n, k, d)$-Code.*

**Beweis**. Lasse im oben konstruierten Code soviel Codeworte weg, bis die Anzahl auf $q^k$ reduziert ist. □

**Bemerkung 11.** *Die so konstruierten Codes sind völlig strukturlos. Die Schranke wird im Allgemeinen nicht besonders gut sein, denn wählt man die Elemente $x$ im Algorithmus gezielt, dann werden pro Schleifeniteration weniger als $Vol(n, d-1)$ Elemente aus $L$ gestrichen. Somit sind mehr Codewörter möglich.*

## 4.2 Gilbert-Varshamov-Schranke

Wir übertragen die eben bewiesene Schranke jetzt auf lineare Codes. Dies geht allerdings nicht so einfach wie oben auf eine konstruktive Art; wir verwenden dazu die folgende „Konstruktions-Idee“:

- Wähle einen Code mit den gewünschten Parametern $(n, k, q)$ gleichverteilt zufällig.
- Zeige, dass dieser zufällige Code mit einer Wahrscheinlichkeit $> 0$ eine Minimaldistanz größer oder gleich $d$ hat.
- Da die Wkt. größer 0 ist existiert so ein Code und der Beweis ist fertig.

Die Existenz-Aussage für einen Code mit den entsprechenden Parametern liefert der folgende Satz:

**Satz 12.** *(Gilbert-Varshamov-Schranke) Sind $n, k, d, q$ gegeben mit*

$$k < n - log_q(Vol_q(n, d-1)),$$

*dann existiert ein linearer $[n, k, d]$-Code über $\mathbb{F}_q$.*

**Beweis**. Fixiere ein $x \neq 0$ aus $\mathbb{F}_q^k$. Wähle eine Generatormatrix $G$ ($n \times k$-Matrix mit Rang $k$) zufällig. Die Vektoren $G \cdot x$ sind dann auch gleichverteilt in $\mathbb{F}_q^n$. Ein solcher Vektor liegt daher mit der folgenden Wahrscheinlichkeit in der Kugel um 0 mit dem Radius $d-1$.

$$\Rightarrow \qquad Pr_G\left[G \cdot x \in B_n(0, d-1)\right] \;=\; \frac{Vol_q(n, d-1)}{q^n}$$

Dies gilt für alle $x \neq 0$. Die Wahrscheinlichkeit der Vereinigung dieser Ereignisse (also dass ein $x$ existiert für das $G \cdot x \in B_n(0, d-1)$) ist nicht größer als die Summe der Wahrscheinlichkeiten der Ereignisse („union bound“ aus der Wkt-Theorie):

$$\begin{aligned}
\Rightarrow \quad Pr_G\left[\exists x \in \mathbb{F}_q^k \setminus \{0\} : G \cdot x \in B_n(0, d-1)\right] &\leq \sum_{x \in \mathbb{F}_q^k \setminus \{0\}} Pr_G\left[G \cdot x \in B_n(0, d-1)\right] \\
&\leq q^k \cdot \frac{Vol_q(n, d-1)}{q^n} \\
&= q^{k-n} Vol_q(n, d-1)
\end{aligned}$$

Für die Wahrscheinlichkeit des komplementären Ereignisses ergibt sich dann:

$$\Rightarrow \quad Pr_G\left[\forall x \in \mathbb{F}_q^k \setminus \{0\} : G \cdot x \notin B_n(0, d-1)\right] \;\geq\; 1 - q^{k-n} Vol_q(n, d-1)$$

Falls diese Wahrscheinlichkeit größer als 0 ist, dann existiert ein linearer Code mit dem Minimalabstand $\geq d$. Diese Wahrscheinlichkeit ist größer als 0, falls

$$1 - q^{k-n} \cdot Vol_q(n, d-1) > 0$$

ist, das ist äquivalent zur geforderten Schranke $k < n - log_q\left(Vol_q(n, d-1)\right)$. □

## 4.3 Singleton Schranke

Zunächst betrachten wir nochmals die Hamming-Schranke: Ist $C$ ein $[n, k, d]$-Code, dann gilt:

$$k \leq n - log_q\left(Vol_q\left(n, \frac{d-1}{2}\right)\right)$$

Diese Schranke ist offensichtlich abhängig von der Alphabetsgröße $q$. Die Singleton-Schranke ist auch eine obere Schranke für die Dimension $k$ eines $[n, k, d]$-Codes. Sie ist allerdings unabhängig von der Größe des zugrunde liegenden Alphabets:

**Satz 13.** *(Singleton-Schranke) Ist $C$ ein $[n, k, d]$-Code, dann gilt:*

$$k \leq n - d + 1.$$

Das heißt, die Singleton-Schranke ist für großes $q$ besser als die Hamming-Schranke. Der recht anschauliche Beweis der Singleton-Schranke verwendet das Punktieren von Codes, das wir zunächst defineren:

**Definition 11.** Es sei $C$ ein $[n, k, d]$-Code über $\mathbb{F}_q$. Der *punktierte Code* $C^\bullet$ zu $C$ entsteht, indem man bei $C$ die letzte Komponente wegläßt.

**Beweis**. (Singleton-Schranke)
Falls $d \geq 2$ dann ist $C^\bullet$ ein $[n-1, k, \bar{d}]$-Code. Die Minimaldistanz kann sich durch Weglassen einer Komponente um maximal 1 reduzieren, daher ist $\bar{d} \in \{d-1, d\}$. Da $d \geq 2$ können durch Punktieren keine zwei Codeworte zusammenfallen, daher ist die Dimension von $C^\bullet$ identisch mit der von $C$.
Diesen Prozess des Punktierens kann man bei $C$ (mindestens) $d-1$ mal wiederholen bis die Minimaldistanz sich auf 1 reduziert hat und sich dann beim Punktieren die Dimension $(k)$ reduzieren könnte.
Es entsteht nach $d-1$ maligem Punktieren also eine $[n-(d-1), k]$-Code. Die Dimension eines Codes kann nicht größer sein als die Dimension des Vektorraums von dem der Code ein Unterraum ist, es folgt: $n-(d-1) \geq k$ und damit $n-d+1 \geq k$. □

# 5 Algebraische Codes

Das Decodieren von allgemeinen linearen Codes ist recht aufwendig, daher versucht man durch das Verwenden von zusätzlichen algebraischen Strukturen (z.B. Skalarprodukte oder Polynomringe) mehr Struktur in die Codes zu integrieren, die dann beim Decodieren ausgenutzt werden kann. Im ersten Beispiel werden Skalarprodukte bei der Codierung verwendet:

## 5.1 Walsh-Hadamard-Code

Der Walsh-Hadamard-Code ist ein linearer Code über einem binären Alphabet, welcher Nachrichtenworte der Länge $k$ auf Codeworte der Länge $2^k - 1$ abbildet. Jede einzelne Komponente des Codeworts entsteht durch die Berechnung des Skalarprodukts des Nachrichtenworts $m \in \mathbb{F}_2^k$ mit einem Vektor $x$ der Länge $k$:

$$c_x := \langle m, x \rangle = \sum_{i=1}^{k} m_i \cdot x_i.$$

Bei den Walsh-Hadamard-Codes werden alle $2^k - 1$ möglichen $x \in \mathbb{F}_2^k \setminus \{0\}$ verwendet.
Das Berechnen der Skalarprodukte ist nichts neues, bei der Multiplikation der Generatormatrix mit dem Nachrichtenwort wird genau dasselbe berechnet. Die Eigenschaften des Skalarprodukts machen hier die Untersuchung der Codes aber einfacher.

**Definition 12.** Sei $k \in \mathbb{N}$, dann ist die Codierung der Nachricht $m \in \mathbb{F}_2^k$ mit dem *Walsh-Hadamard-Codes* gegeben durch:

$$WH_k(m) = (\langle m, x \rangle)_{x \in \mathbb{F}_2^k \setminus \{0\}}$$

Das besondere am Walsh-Hadamard-Code ist, dass alle Codeworte außer $(0, \ldots, 0)$ das Hamminggewicht $2^{k-1}$ haben.
Betrachten wir uns zunächst die Generatormatrix des Walsh-Hadamard-Codes. Ihre Zeilen sind alle möglichen Worte aus $\mathbb{F}_2^k$ außer dem 0-Wort:

$$G_{WH} = \left.\begin{pmatrix} 0 & \ldots & 1 \\ & \vdots & \\ 1 & \ldots & 1 \end{pmatrix}\right\} 2^k - 1 \text{ Zeilen (alle möglichen Worte außer dem 0-Wort)}$$

**Lemma 14.** *Der Walsh-Hadamard-Codes $WH_k$ ist ein $[2^k - 1, k, 2^{k-1}]$-Code.*

**Beweis**. Die Menge $\mathbb{F}_2^k \setminus \{0\}$ hat $2^k - 1$ Elemente, daher ergeben sich direkt die ersten beiden Parameter von $WH_k$. Durch die Codierung mit der Generatormatrix ist auch klar, dass $WH_k$ ein linearer Code ist, es bleibt also nur noch z.z. dass die Minimaldistanz $2^{k-1}$ ist.
Betrachten wir eine Nachricht $m \neq (0, \ldots, 0)$, o.B.d.A. sei $m = (1, \bar{m})$ für ein $\bar{m} \in \mathbb{F}_2^{k-1}$. Sei $\bar{x} \in \mathbb{F}_2^{k-1}$ beliebig gewählt, dann ergeben sich für die beiden Vektoren $x_0 := (0, \bar{x})$ und $x_1 := (1, \bar{x})$ bei der Berechnung der Skalarprodukte mit $m$ die beiden verschiedenen Werte:

$$\begin{aligned} m_{x_0} &= \langle m, x_0 \rangle = 0 + \langle \bar{m}, \bar{x} \rangle \\ m_{x_1} &= \langle m, x_1 \rangle = 1 + \langle \bar{m}, \bar{x} \rangle = m_{x_0} + 1. \end{aligned}$$

Bei der Codierung von $m$ sind daher $2^{k-1}$ der Komponenten 1 und $2^{k-1} - 1$ der Komponenten 0 (Weglassen des Nullvektors bei $x$). Alle Codeworte (ausser dem Nullwort) haben das Gewicht $2^{k-1}$. □

Betrachten wir nochmals die Generatormatrix des Walsh-Hadamard-Codes. Nach Konstruktion ist dies die Transponierte der Prüfmatrix $H_{Ham}$ des Hamming Codes:

$$G_{WH} = H_{Ham}^t.$$

Wie sieht nun die Prüfmatrix des Walsh-Hadamard-Codes aus? Für die Generator- und Prüfmatrix des Hamming-Cods gilt:

$$H_{Ham} \cdot G_{Ham} = 0.$$

Transponieren der Matrizen liefert $G_{Ham}^t \cdot H_{Ham}^t = 0$ und damit ist $G_{Ham}^t$ eine Prüfmatrix des Walsh-Hadamard-Codes.
Damit ergibt sich eine direkte Beziehung zwischen den Prüfmatrizen und den Generatormatrizen des Hamming Codes und des Walsh-Hadamard-Codes. Dies ist ein allgemeines Konzept, das sich im Begriff des dualen Codes widerspiegelt:

## 5.2 Dualer Code

Zu jedem linearen Code $C$ gibt es einen dualen Code (Dualcode) $C^\perp$, der selbst ein linearer Code ist. Dieser wird mit Hilfe des Skalarprodukts

$$\begin{aligned} \langle \cdot, \cdot \rangle \colon \mathbb{F}_q^n \times \mathbb{F}_q^n &\to \mathbb{F}_q \\ \langle x, y \rangle &:= \sum_{i=1}^{n} x_i y_i \end{aligned}$$

definiert.

**Definition 13.** Es sei $C$ ein $[n, k]$-Code über $\mathbb{F}_q$. Dann ist der *duale Code* definiert durch:

$$C^\perp := \left\{ x \in \mathbb{F}_q^n \mid \langle x, c \rangle = 0 \qquad \forall c \in C \right\}$$

Die Codeworte des dualen Codes sind alle Wörter aus $\mathbb{F}_q^n$, die zu allen Codewörtern aus $C$ dual sind.
Mit denselben Betrachtungen wie beim Walsh-Hadamard-Code ergibt sich die folgende Beziehung zwischen den Generator- und Prüfmatrizen.

**Lemma 15.** *Es sei $C$ ein $[n,k]$-Code mit Generatormatrix $G$ und Prüfmatrix $H$. Dann gilt für den dualen Code:*
*Die Generatormatrix von $C^\perp$ ist $H^t$, die Prüfmatrix von $C^\perp$ ist $G^t$.*

**Vorsicht:** Ist das Skalarprodukt (bei den reellen Zahlen) von zwei Vektoren 0, so bedeutet die anschaulich, dass diese (reellwertigen) Vektoren senkrecht aufeinander stehen. Über $\mathbb{F}_q$ stimmt diese Anschauung nicht. Es gibt dort Vektoren die zu sich selbst dual sind, z.B. $(1,0,0,1) \in \mathbb{F}_2^4$. Es gibt auch Untervektorräume die zu sich selbst dual sind; die entsprechenden Codes heissen dann selbstdual.

## 5.3 Reed-Solomon-Code (RS-Code)

Ein anderes Beispiel für algebraische Strukturen in der Codierungstheorie sind Polynomringe. Bei Reed-Solomon-Codes wird eine Nachricht $m$ zunächst als die Koeffizienten eines Polynoms interpretiert und dann codiert, indem das Polynoms an $n$ festen Stützstellen ausgewertet wird. Das Polynom ist vom Grad $\leq k-1$ über $\mathbb{F}_q$.
Für das Decodieren ergeben sich die Koeffizienten des Polynoms wieder durch Interpolation bzw. als Lösung eines LGS.

**Definition 14.** Sei $\mathbb{F}_q$ ein endlicher Körper, $k \leq q$, $P_k$ die Menge aller Polynome in $\mathbb{F}_q[x]$ vom Grad $< k$. Dann ist der $RS_{q,k}$ gegeben durch die Codierungsabbildung

$$\begin{aligned} RS_{q,k}\colon P_k &\to \mathbb{F}_q^q \\ m &\mapsto (m(a))_{a\in\mathbb{F}_q}. \end{aligned}$$

**Satz 16.** *Das Bild der Codierabbildung $RS_{q,k}$ ist ein linearer $[q,k,q-k+1]$-Code über $\mathbb{F}_q$.*

**Beweis**. Die Länge $q$ und die Dimension $k$ ergeben sich direkt aus der Definition, da $\mathbb{F}_q$ genau $q$ Elemente hat und $P_k$ ein Vektorraum der Dimension $k$ über $\mathbb{F}_q$ ist.
Zu zeigen sind noch die Linearität und dass die Minimaldistanz $d_{min} = q-k+1$ ist.

**Linearität:** Seien $f(x), g(x)$ die Polynome zu zwei Nachrichtenwörtern mit den dazugehörigen Codewörtern $(f(\xi_i)), (g(\xi_i)) \in \mathbb{F}_q^q$.
Zu zeigen: $\alpha\cdot(f(\xi_i)) + \beta\cdot(g(\xi_i)) \in RS$ für $\alpha,\beta \in \mathbb{F}_q$
Das Nachrichtenpolynom $\alpha\cdot f(x) + \beta\cdot g(x)$ ist ein Polynom vom Grad $\leq k-1$ und wird auf das gesuchte Codewort abgebildet (da Auswerten homomorph ist).

**Minimaldistanz:** Sei $f(x) \neq 0$ ein Polynom vom Grad $\leq k-1$.
Es hat daher maximal $k-1$ Nullstellen.
$\Rightarrow f(\xi_i) \neq 0$ für mindestens $q-(k-1)$ verschiedene $\xi_i \in \mathbb{F}_q$
$\Rightarrow d_{min} \geq q-k+1$
Die Singleton-Schranke liefert $d_{min} \leq q-k+1$ und daher die exakte Minimaldistanz. $\square$

Eine einfache Modifikation der RS-Codes ist es nicht alle Auswertungsstellen aus $\mathbb{F}_q$ zu verwenden. Um die Parameter eines solchen Codes zu bestimmen gehen wir umgekehrt vor — wir lassen beim RS-Code Auswertungsstellen weg. Das ist das bereits bekannte Punktieren eines Codes.

### 5.3.1 Punktierter RS-Code

Sei $C$ ein RS-Code der Dimension $k$ über $\mathbb{F}_q$ und $n \geq k$. Punktieren wir $C$ nun $r := q - n$ mal so liefert dies einen $[\underbrace{q - (q-n)}_{n}, k]$-Code mit $d_{min} \geq q - k + 1 - (q - n) = n - k + 1$.
Wegen der Singleton-Schranke gilt $d_{min} \leq q - k + 1 - (q - n) = n - k + 1$ und damit ist auch für punktierte RS-Codes die Minimaldistanz eindeutig bestimmt. Das beweist:

**Satz 17.** *Sei $C$ ein $[q, k, q - k + 1]$-Reed-Solomon-Code über $\mathbb{F}_q$ und $n \geq k$. Dann ist der $q - n$ fach punktierte Code ein $[n, k, n - k + 1]$-Code.*

Diese Codes erfüllen alle die Singleton-Schranke exakt. Dies ist eine eher seltene Eigenschaft bei linearen Codes, Codes die dies erfÃŒllen heissen MDS-Codes:

**Definition 15.** Ein Code $C$ der die Singleton-Schranke mit Gleichheit erfÃŒllt heißt *maximum distance seperable* oder *MDS-Code.*

Mit demselben Beweis wie oben bei RS-Codes sieht man auch:

**Lemma 18.** *Punktieren eines MDS-Codes liefert wieder einen MDS-Code.*

### 5.3.2 Verallgemeinerter RS-Code (GRS-Code)

Eine weitere, nicht ganz so naheliegende Verallgemeinerung von RS-Codes ist es die verschiedenen Komponenten der Codeworte noch zusätzlich mit verschiedenen Körperelementen zu multiplizieren. Der zusätzliche Aufwand beim Codieren und Decodieren ist sehr gering, man gewinnt allerdings eine zusätzliche algebraische Eigenschaft: Der Dualcode eines (derart) erweiterten RS-Codes ist wieder ein erweiterter RS-Code.

**Definition 16.** Sei $\mathbb{F}_q$ ein endlicher Körper, $\xi = (\xi_1, \ldots, \xi_n) \in \mathbb{F}_q^n$ mit $\xi_1, \ldots, \xi_n$ paarweise verschieden und $\mathbf{v} = (v_1, \ldots, v_n) \in (\mathbb{F}_q \setminus \{0\})^n$. Sei $P_k$ die Menge aller Polynome in $\mathbb{F}_q[X]$ vom Grad $< k$. Der $q$-äre *verallgemeinerte Reed Solomon Code* $GRS_{n,k,\xi,\mathbf{v}}$ ist gegeben durch die Codierungsabbildung

$$\begin{array}{rccc} GRS_{n,k,\xi,\mathbf{v}}: & P_k & \to & \mathbb{F}_q^n \\ & f(x) & \mapsto & (v_i f(\xi_i))_{i=1,\ldots,n} \end{array}$$

**Satz 19.** *Der verallgemeinerte Reed-Solomon-Code*

$$GRS_{\xi,v}(f(x)) := (v_1 \cdot f(\xi_1), \ldots, v_n \cdot f(\xi_n))$$

*ist ein $[n, k, n - k + 1]$-Code.*

**Beweis**. Der Beweis ist sehr ähnlich dem des RS-Codes. Die verallgemeinerten Codes sind lineare $[n, k]$-Codes, wie die RS-Codes. Der Beweis für die Minimaldistanz ist auch ähnlich dem des RS-Codes. Jedes Polynom $f(x) \neq 0$ vom Grad $k-1$ hat höchstens $k-1$ viele Nullstellen. Also hat $f(x)$ mindestens $n-(k-1)$ viele Nichtnullstellen. Ist $f(\xi_i) \neq 0$, so ist auch $v_i \cdot f(\xi_i) \neq 0$, da $v_i \neq 0$ und $\mathbb{F}_q$ ein Körper ist. Damit ist $wgt(GRS_{v,\xi}(f(x))) \geq n-k+1$. Wegen der Singleton-Schranke ist die Minimaldistanz $d(GRS_{\xi,v}(f(x))) = n-k+1$. □

**Lemma 20.** *Der Duale Code des $GRS_{\xi,v,k}$ ist der $GRS_{\xi,w,n-k}$ mit*

$$w_i = \frac{1}{v_i \cdot \prod_{j \neq i}(\xi_i - \xi_j)}.$$

**Beweis**. Es ist zu zeigen, dass die Codeworte aufeinander orthogonal sind:
$\forall c \in GRS_{\xi,v,k}$ und $\forall h \in GRS_{\xi,w,n-k}$ muss $\langle h, c \rangle = 0$ gelten.
Sei $c = GRS_{\xi,v,n}(f)$ mit $deg(f) \leq k-1$ und $h = GRS_{\xi,w,n-k}(g)$ mit $deg(g) \leq n-k-1$.

$$\begin{aligned}
\langle h, c \rangle = \sum_{i=1}^{n} h_i c_i &= \sum_{i=1}^{n} w_i g(\xi_i) \cdot v_i f(\xi_i) \\
&= \sum_{i=1}^{n} \frac{1}{v_i \cdot \prod_{j \neq i}(\xi_i - \xi_j)} g(\xi_i) \cdot v_i f(\xi_i) \\
&= \sum_{i=1}^{n} \frac{1}{\prod_{j \neq i}(\xi_i - \xi_j)} g(\xi_i) \cdot f(\xi_i) \\
&= \sum_{i=1}^{n} \frac{1}{\prod_{j \neq i}(\xi_i - \xi_j)} (g \cdot f)(\xi_i)
\end{aligned}$$

$f \cdot g$ ist ein Polynom vom Grad $\leq k-1+n-k-1 = n-2$. Da der Grad kleiner als $n-1$ ist, kann $f \cdot g$ mithilfe der Lagrange Interpolation zu

$$(g \cdot f)(x) = \sum_{i=1}^{n} \underbrace{\left( \prod_{j \neq i} \frac{x - \xi_j}{\xi_i - \xi_j} \right)}_{\text{Grad } n-1} (f \cdot g)(\xi_i)$$

entwickelt werden. Der Leitkoeffizient ist $\frac{1}{\prod_{j \neq i}(\xi_i - \xi_j)}$, was auch der $n-1$-te Koeffizent von $(f \cdot g)(x)$ ist. Da jedoch der Grad von $(f \cdot g) \leq n-2$ ist, ist der $n-1$-te Koeffizient $= 0$.
Somit ist auch $\langle h, c \rangle = 0$. □

### 5.3.3 Decodieren des GRS-Codes

Sind die Fehlerstellen bekannt, so kann mit den richtig empfangenen Koeffizienten per Interpolation direkt das Nachrichtenpolynom berechnet werden. Werden einige Koeffizienten z.B. nicht empfangen (Kratzer auf einer CD), so kann dieses Verfahren funktionieren. Im Allgemeinen werden die Fehlerstellen aber nicht bekannt sein. In diesem Fall müssen zunächst die falschen Koeffizienten gefunden werden. Dabei spielt das Fehlerstellenpolynom $e(x)$ (Error Locator Polynom) eine zentrale Rolle.

Für ein Codewort $c$ habe das fehlerbehaftete Codewort $\tilde{c}$ genau $t$ Fehler, so dass $d(\tilde{c}, c) = t$ ist. Die Menge der Indizes an welchen $\tilde{c}$ von $c$ abweicht sei $I := \{i_1, \ldots, i_t\}$; die Codewortpositionen $i$ entsprechen dabei den Auswertungsstellen $\xi_i$. Das Fehlerstellenpolynom wird folgendermaßen definiert:

$$e(x) := \prod_{i \in I} (x - \xi_i)$$

Diese Polynom ist genau an den Fehlstellen Null. Die $t$ unbekannten Fehlerstellen, also die richtige der $\binom{n}{t}$ möglichen Fehlermengen, werden durch dieses Polynom, als durch die $t$ unbekannten Koeffizienten des Polynoms, eindeutig beschrieben. Diese sehr kompakte Darstellung der „Fehlerstellen“ ist der Kern der effizienten Decodierung.
Das Codewort zum Nachrichten-Polynom $f(x)$ vom Grad $\leq k - 1$ sei

$$c_i = (v_i \cdot f(\xi_i))_{i=1,\ldots,n}.$$

Das fehlerbehaftete empfangene Wort $\tilde{c}$ enthält $t$ Fehlern, wobei $t \leq \frac{d-1}{2}$ ist. Betrachten wir nun das Polynom

$$p(x) = f(x) \cdot e(x)$$

und werten dieses Polynom an den Stellen $\xi_1, \ldots, \xi_n$ aus:

$$p(\xi_i) = \frac{\tilde{c}_i}{v_i} \cdot e(\xi_i) = \frac{\tilde{c}_i}{v_i} \cdot \prod_{i \in I} (x - \xi_i) \tag{2}$$

Diese Gleichung heißt Key Equation. Zunächst würden in dieser Gleichung nicht die empfangenen Koeffizienten $\tilde{c}_i$, sonder die unbekannten Werte $c_i$ stehen. Durch die Definition von $e(x)$ gelten die Gleichungen aber auch für die fehlerbehafteten $\tilde{c}_i$. Dies ist, wie der Name sagt, der Schlüssel für die Decodierung.

**Lemma 21.** *Für alle $i = 1, \ldots, n$ gilt die Key Equation (2).*

**Beweis**. Wir unterscheiden die beiden Fälle $i \in I$ und $i \notin I$:

***i* $\notin$ *I*:** Falls $i$ keine Fehlerstelle ist, dann sind $c_i$ und $\tilde{c}_i$ identisch, also $\tilde{c}_i = v_i \cdot f(\xi_i)$. Somit gilt $p(\xi_i) = \dfrac{\tilde{c}_i}{v_i} \cdot e(\xi_i)$.

***i* $\in$ *I*:** Falls $i$ jedoch eine Fehlerstelle ist, dann ist $e(\xi_i) = 0$. Das heißt

$$p(\xi_i) = \frac{\tilde{c}_i}{v_i} \cdot e(\xi_i) = 0.$$

□

Durch diese $n$ Gleichungen versuchen wir jetzt die unbekannten Polynome $p(x)$ und $e(x)$ zu bestimmen:

- Setzen wir dazu $e(x)$ mit $t$ unbestimmten Koeffizienten an (ist ein normiertes Polynom vom Grad $t$).
- Das Polynom $p(x)$ hat $\deg(p) = \deg(f) + \deg(e) \leq k + t - 1$ und kann mit $k + t$ unbestimmten Koeffizienten angesetzt werden.

- Die Schlüsselgleichungen liefern mit diesem Ansatz $n$ lineare Gleichungen in $k+t+t$ Unbestimmten (die Werte $\tilde{c}_i, v_i, \xi_i$ für $i = 1, \ldots, n$ sind bekannt, jedes $\xi_i$ liefert eine Gleichung).

- Man kann das LGS lösen, falls $n \geq k + 2 \cdot t$; da laut Voraussetzung $t \leq \frac{d-1}{2}$ ist $k + 2 \cdot t \leq k + d - 1 = k + n - k = n$. Eine Lösung des inhomogenen LGS existiert auch, nämlich das richtiges Codewort.

Ist nun eine Lösung $\tilde{p}(x), \tilde{e}(x)$ des LGS gefunden, dann gilt:

$$\begin{aligned} & \tilde{p}(x) = f(x) \cdot \tilde{e}(x) \\ \Rightarrow \quad & f(x) = \frac{\tilde{p}(x)}{\tilde{e}(x)} \qquad \text{ist die Nachricht.} \end{aligned}$$

Es muss noch gezeigt werden, dass jede Lösung des LGS auch die richtige Nachricht liefert. Dies sieht man durch entsprechendes Nachrechnen. (Betrachtet man die Anzahl der Nullstellen von $\tilde{p}(x) - f(x) \cdot \tilde{e}(x)$ (mindestens alle $\xi_i$ der korrekt empfangenen Stellen), so ist klar, dass $\tilde{p}(x) - f(x) \cdot \tilde{e}(x) = 0(x)$.)
Der Aufwand dieses Algorithmus wird dominiert vom Lösen eines $n \times n$ LGS, benötigt also maximal $n^3$ Schritte. Damit haben wir folgendes gezeigt:

**Theorem 22.** *Es existiert ein effizienter Decoder für $GRS_{n,k,\xi,v}$ welcher $\frac{n-k}{2} = \frac{d-1}{2}$ Fehler korrigiert.*

## 5.4 Goppa Codes

Eine weitere Klasse von polynomialen Codes sind die Goppa Codes. Um sie definieren zu können benötigen wir noch ein paar Eigenschaften von Pollynomen über endlichen Erweiterungskörpern:

- Sei $g(X)$ ein Polynom aus $\mathbb{F}_q[X]$, dann ist $\mathbb{F}_q[X]/(g(X))$ ist ein modularer Polynomring;

- die Elemente sind Restklassen von Polynomen modulo $g(X)$;

- ein Repräsentantensystem sind die Polynome vom Grad kleiner $\deg(g(x))$;

- ein Polynom $f(X) \in \mathbb{F}_q[X]/(g(X))$ besitzt genau dann ein Inverses in dem modularen Polynomring, wenn $ggT(f(X), g(X)) = 1$, also wenn $f(X)$ und $g(X)$ teilerfremd sind.
  Das Inverse kann über den erweiterten euklidschen Algorithmus berechnet werden.

- Falls $g(\alpha) \neq 0$, dann sind $X - \alpha$ und $g(X)$ teilerfremd.
  (Sonst wäre $g(X) = (X - \alpha)h(X)$ und $\alpha$ wäre eine Nullstelle von $g(X)$.)

### 5.4.1 Parameter der Goppa Codes

Mit diesen Hilfsmitteln können wir jetzt den Goppa Code definieren:

**Definition 17.** Sei $\mathbb{F}_q$ ein endlicher Körper und $\mathbb{F}_{q^m}$ ein Erweiterungskörper. Sei $g(x) \in \mathbb{F}_{q^m}[x]$ ein Polynom vom Grad $t$ und $\alpha_i = (\alpha_1, \ldots, \alpha_n) \in (\mathbb{F}_{q^m}^n)$ paarweise verschieden mit $g(\alpha_i) \neq 0$ für $i = 1, \ldots, n$.
Der *Goppa Code* $\Gamma(\alpha, g(x))$ ist die Menge aller Vektoren $c = (c_1, \ldots, c_n) \in \mathbb{F}_q^n$ für die

$$\sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} = 0 \pmod{g(x)}$$

gilt. (Beachte: Hier steht **nicht** „$c \in \mathbb{F}_{q^m}^n$“.)

Der Code wird nicht über eine Generatormatrix, sondern (ähnlich wie der Hamming Code) über Prüfgleichungen definiert.

**Satz 23.** *Sei*

$$\Gamma(\alpha, g(x)) = \left\{ c = (c_1, \ldots, c_n) \in \mathbb{F}_q^n \;\middle|\; \sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} = 0 \right\}.$$

*der Goppa Code zu $\alpha \in \mathbb{F}_{q^m}^n, g(x) \in \mathbb{F}_{q^m}[x]$ vom Grad $t$.*
*Der Code $\Gamma(\alpha, g)$ ist ein $[n, n-k, d]$-Code der Dimension $n - k \geq n - t \cdot m$ und der Minimaldistanz $d \geq t + 1$.*

**Beweis**. Betrachten wir das Polynom

$$s_c(x) := \sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} = \sum_{j=0}^{t-1} s_j(c) \cdot x^j \pmod{g(x)}.$$

Damit gilt die Äquivaenz:

$$s_c(x) = 0 \quad \Leftrightarrow \quad s_0 = \ldots = s_{t-1} = 0 \quad \Leftrightarrow \quad c \in \Gamma(\alpha, g(x)).$$

Es müssen die folgenden drei Punkte betrachtet werden:

1. Linearität:

   Ist $s_{c_1}(x) = 0$ und $s_{c_2}(x) = 0$, dann ist auch $s_{c_1+c_2}(x) = 0$, dies liefert direkt die Linearität von $\Gamma(\alpha, g(x))$.

2. Dimension $n - k \geq n - t \cdot m$:

   $s_c(x) \pmod{g(x)}$ ist ein Polynom in $\mathbb{F}_{q^m}$ vom Grand $< t$. Es müssen also die (maximal) $t$ Gleichungen $s_0 = \ldots = s_{t-1} = 0$ in $\mathbb{F}_{q^m}$ erfüllt sein.

   Jede dieser Gleichungen kann durch $m$ viele linearen Gleichungen über $\mathbb{F}_q$ beschrieben werden.

   Insgesamt lässt sich $\Gamma(\alpha, g(x))$ also durch $m \cdot t$ Gleichungen über $\mathbb{F}_q$ beschreiben. Diese Gleichungen können linear abhängig sind, reduziere sie auf ein maximales linear unabhängiges System mit $k \leq m \cdot t$ Gleichungen. Dies liefert eine $(k \times n)-$Prüfmatrix $H$ mit $k \leq m \cdot t$.

   Die Dimension des Goppa Codes ist also $\dim(\Gamma(\alpha, g(x))) = n - k \geq n - t \cdot m$ wie behauptet.

3. Minimaldistanz $d_{min} \geq t+1$:

   Sei $0 \neq (c_1, \ldots, c_n) \in \mathbb{F}_q^n$ ein Wort mit $c_i \neq 0$ genau für $i \in I$ und $|I| =: l \leq t$.

   Wir wollen nun zeigen, dass $s_c(x) \neq 0$ gilt:

   Setze dazu

$$e(x) := \prod_{j \in I} (x - \alpha_j).$$

   Dies ist ein Polynom vom Grad $l \leq t$ (da $|I| = l$). Betrachte nun

$$\begin{aligned} s_c(x) \cdot e(x) &= \sum_{i=1}^{n} c_i \cdot \frac{1}{x - \alpha_i} \cdot \prod_{j \in I} (x - \alpha_j) \\ &= \sum_{i \in I} c_i \cdot \prod_{\substack{j \in I, \\ j \neq i}} (x - \alpha_j) \qquad \text{da } c_i = 0 \text{ für } c_i \notin I \end{aligned}$$

   Jeder Summand ist ein Polynom vom Grad $l - 1 \leq t - 1$, also ist auch die Summe $s_c(x) \cdot e(x)$ ein Polynom vom Grad $\leq l - 1 \leq t - 1$.

   Daher können wir $s_c(x) \cdot e(x)$ als Polynom in $\mathbb{F}_{q^n}[x]$ auffassen, d.h. der Modulus $g(x)$ wird nicht angewandt, da $\deg(g(x)) > \deg(s_c(x) \cdot e(x))$.

| | | | |
|---|---|---|---|
| Sei $i_0 \in I$ | $\Rightarrow$ | $s_c(\alpha_{i_0}) \cdot e(\alpha_{i_0}) = c_{i_0} \cdot \prod_{\substack{j \in I, \\ j \neq i_0}} \underbrace{(\alpha_{i_0} - \alpha_j)}_{\neq 0}$ | alle anderen Summanden sind Null, da der Faktor $(\alpha_{i_0} - \alpha_{i_0})$ enthalten ist. |
| | $\Rightarrow$ | $s_c(x) \cdot e(x) \neq 0$ | als Polynom |
| | $\Rightarrow$ | $s_c(x) \neq 0$ | als Polynom |
| | $\Rightarrow$ | $c \notin \Gamma(\alpha, g(z))$ | für alle $c$ vom Gewicht $\leq t$ |
| | $\Rightarrow$ | $d_{min} > t$ | |

□

Die Konstruktion eines Codes über $\mathbb{F}_q$ indem Prüfgleichungen durch Restklassen von Polynomen über $\mathbb{F}_{q^n}$ definiert werden verwendet einiges an algebraischen Strukturen. An einem Beispiel soll die Konstruktion daher nochmals durchgeführt weden:

**Beispiel.** Wir konstruieren einen binärem Goppa Code, d. h. $q := 2$. Als Grad der Körpererweiterung wählen wir $m := 3$, der Körper $\mathbb{F}_{2^3}$ ist

$$\mathbb{F}_{2^3} = \{0, 1, a, a+1, a^2, a^2+1, a^2+a, a^2+a+1\}, \text{ wobei } a^3 + a + 1 = 0 \text{ gilt.}$$

Als Polynom wählen wir $g(x) := a \cdot x^2 + a + 1$, dieses Polynom hat nur eine Nullstelle, nämlich $a \in \mathbb{F}_{2^3}$. Von den 7 Nichtnullstellen von $g(x)$ wählen wir uns die folgenden 5 aus (in der angegebenen Reihenfolge):

$$\alpha = (\alpha_1, \ldots, \alpha_5) := (0, a^2, a^2+a, 1, a^2+1).$$

Dazu berechnen wir die fünf Polynome $\frac{1}{x - \alpha_i} \mod g(x)$, dies sind dann

$$\Big((a^2+a+1) \cdot x, (a^2+1) \cdot x + a, (a+1) \cdot x + 1, a \cdot x + a, (a^2+a) \cdot x + a + 1\Big).$$

Das Syndrompolynom zu einem Wort $c = (c_1, \ldots, c_5) \in \mathbb{F}_2^5$ ist dann:

$$\begin{aligned} s_c(x) &= c_1 \cdot (a^2 + a + 1) \cdot x + c_2 \cdot ((a^2 + 1) \cdot x + a) + c_3 \cdot ((a + 1) \cdot x + 1) + \\ &\quad c_4 \cdot (a \cdot x + a) + c_5 \cdot ((a^2 + a) \cdot x + a + 1) \\ &= \Big((a^2 + a + 1)c_1 + (a^2 + 1)c_2 + (a + 1)c_3 + a \cdot c_4 + (a^2 + a)c_5\Big) \cdot x + \\ &\quad a \cdot c_2 + c_3 + a \cdot c_4 + (a + 1)c_5. \end{aligned}$$

Für Codeworte $c = (c_1, \ldots, c_5) \in \mathbb{F}_2^5$ müssen die beiden Gleichungen

$$\begin{aligned} 0 &= (a^2 + a + 1)c_1 + (a^2 + 1)c_2 + (a + 1)c_3 + a \cdot c_4 + (a^2 + a)c_5 \text{ und} \\ 0 &= a \cdot c_2 + c_3 + a \cdot c_4 + (a + 1)c_5 \end{aligned}$$

gelten.
Da $c_1, \ldots c_5$ $in \mathbb{F}_2$ müssen die Summen der $\mathbb{F}_{2^3}$ Werte in jeder Komponente $(1, a, a^2)$ Null ergeben. Es ergeben sich die 6 Gleichungen ÃŒber $F_2$:

$$\begin{aligned} 0 = &\quad c_1 + c_2 + c_3 &&\text{Gleichung zu „1“} \\ 0 = &\quad c_1 + c_3 + c_4 + c_5 &&\text{Gleichung zu „}a\text{“} \\ 0 = &\quad c_1 + c_2 + c_5 &&\text{Gleichung zu „}a^2\text{“} \\ 0 = &\quad c_3 + c_5 &&\text{Gleichung zu „1“} \\ 0 = &\quad c_2 + c_4 + c_5 &&\text{Gleichung zu „}a\text{“} \\ 0 = &\quad 0 &&\text{Gleichung zu „}a^2\text{“} \end{aligned}$$

Schreiben wir diese 6 Gleichungen in Form einer Matrix:

$$\tilde{H}^t = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

und reduzieren die 6 Zeilen auf ein linear unabhängiges System, so erhalten wir die folgende Prüfmatrix des Goppacodes

$$H^t = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Eine Generatormatrix des $[5, 2, 3]$-Goppa Codes ist dann:

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

Die Minimaldistanz ist nach dem Satz von oben $\geq t + 1 = 3$, dass sie hier nicht größer ist sieht man $G$ an. Im allgemeinen kann die Minimaldistanz aber größer sein als $t + 1$.

### 5.4.2 Decodierung der Goppa Codes

Für eine effiziente Fehlerkorrektur wird wie bei Reed-Solom Codes die Key Equation verwendet. Mit ihrer Hilfe kann der Fehlervektor (error locator) gefunden werden, woraus die Positionen der Fehler berechnet werden. Ist der Fehlervektor bekannt, kann durch Lösen von linearen Gleichungen das ursprüngliche Codewort gefunden werden.
Sei $\tilde{c} = c + e \in \mathbb{F}_q^n$ ein empfangenes Wort, $e$ der Fehlervektor vom Gewicht $wgt(e) = l \leq \frac{t}{2}$ (mit $l > 0$) und $I \subset \{1, \ldots, n\}$ die Menge der Fehlerstellen.
Es gilt für das Syndrompolynom zu $\tilde{c}$:

$$\begin{aligned} s_{\tilde{c}}(x) &= \overbrace{s_c(x)}^{=0} + s_e(x) = s_e(x) \\ &= \sum_{i=1}^{n} e_i \cdot \frac{1}{x - \alpha_i} \\ &= \sum_{i \in I} e_i \cdot \frac{1}{x - \alpha_i} \quad \text{, da } e_i = 0 \text{ für alle } i \notin I \end{aligned}$$

Wir definieren nun das Fehlerstellen-Polynom:

$$e(x) := \prod_{i \in I} (x - \alpha_i)$$

und betrachten dann die Schlüsselgleichung:

$$\begin{aligned} p(x) &:= s_{\tilde{c}}(x) \cdot e(x) \\ &= s_e(x) \cdot e(x) \\ &= \sum_{i \in I} e_i \frac{\prod_{j \in I}(x - \alpha_j)}{x - \alpha_i} \\ &= \sum_{i \in I} e_i \cdot \prod_{\substack{j \in I, \\ j \neq i}} (x - \alpha_j). \end{aligned}$$

Jeder Summand $\prod_{\substack{j \in I, \\ j \neq i}} (x - \alpha_j)$ ist ein Polynom vom Grad $l - 1$, also ist die Summe $p(x)$ ein Polynom vom Grad $\leq l - 1$.
Das Syndrompolynom $s_{\tilde{c}}(x)$ kann bei gegebenen $\tilde{c}$ effizient berechnet werden.
Setzen wir nun die Schlüsselgleichung mit unbekanntem $p(x)$ (Grad $\leq l - 1$, also $l$ Unbestimmte) und unbekanntem $e(x)$ (normiertes Polynom vom Grad $l$, also $l$ Unbestimmte) an, dann ergeben sich $t$ Gleichungen in $2 \cdot l$ Unbestimmten. (Die Rechnung wird modulo $g(x)$ mit $\deg(g(x) = t$ durchgefÃŒhrt, also ergeben sich beim Koeffizientenvergleich $t$ Gleichungen.)
Da $l \leq \frac{t}{2}$ finden wir dadurch also mindestens so viel Gleichungen wie Unbestimmte und können das lineare Gleichungssystem Lösen, da mindestens eine von Null verschiedene Lösung, nämlich $s_e(x) \cdot e(x)$, existiert. Diese Lösung liefert $e(x)$ und damit auch das gesuchte Codewort $c$. (Für die Korrektheit des Decodieralgorithmus muss man noch zeigen dass jede gefundene Lösung des LGS zum richtigen Codewort führt, dies wird hier übersprungen.)

Beim konkreten Decodieren kennen wir die Anzahl $l$ der aufgetretenen Fehler nicht. Wir versuchen das Decodieren daher für $l = \frac{t}{2}$ und reduzieren $l$ bis der Algorithmus keine Lösung mehr findet. Die letzte gefundene Lösung ist die gesuchte.

**Satz 24.** *Es existiert ein effizienter Decoder für $\Gamma(\alpha, g)$ welcher bis zu $l \leq \frac{t}{2} = \frac{d-1}{2}$ Fehler korrigiert.*

## 5.5 Das McEliece Kryptosystem

Das Effiziente Decodieren von Goppa Codes und die Schwierigkeit des Decodierens von „allgemeinen Codes“ ist die Grundidee für eine Cryptosystem, das schon 1978 von Robert J. McEliece vorgeschlagen wurde. Diese System gilt als sicher, wird allerdings recht selten wirklich eingesetzt: Um dasselbe Sicherheitsniveau wie bei RSA zu erreichen müssen sehr lange Schlüssel verwendet werden, sie sind etwa einen Faktor 1000 länger als die entsprechenden RSA-Schlüssel.

### 5.5.1 Erzeugen der Schlüssel

Um das System aufzusetzen gehen wir folgendermaßen vor:

- wir wählen einen $(n, k, d)$-Goppa Code $\Gamma$ mit Generatormatrix $G \in \mathbb{F}_2^{n \times k}$;
- wir wählen eine (zufällige) invertierbare Matrix $S \in \mathbb{F}_2^{k \times k}$;
- wir wählen eine (zufällige) Permutationsmatrix $P \in \mathbb{F}_2^{n \times n}$;
- wir berechnen $\hat{G} := P \cdot G \cdot S \in \mathbb{F}_2^{n \times k}$.

Der öffentliche Schlüssel besteht aus $\hat{G}$ und den Parametern $(n, k, d)$.
Der geheime Schlüssel besteht aus $(S, G, P)$.

### 5.5.2 Verschlüsseln von Nachrichten

Um eine Nachricht $m \in \mathbb{F}_2^k$ zu verschlüsseln wird ein zufälliger „Fehler-Vektor“ $e \in \mathbb{F}_2^n$ mit Gewicht $d$ gewählt (Zeilenvektor). Die Nachricht wird dann mit $\hat{G}$ codiert und mit $e$ verfälscht:

$$c := \hat{G} \cdot m + e.$$

### 5.5.3 Entschlüsseln von Chiffraten

Um das Chiffrat $c \in \mathbb{F}_2^n$ zu entschlüsseln wird sukzessive vorgegangen, entsprechend der Zerlegung $\hat{G} := P \cdot G \cdot S$:

- Wir berechnen $c' := P^{-1} \cdot c$;
  dann ist $c'$ das um $P^{-1} \cdot e$ verfälschte Codewort von $\Gamma$ zur Nachricht $S \cdot m$. Da $P$ eine Permutationsmatix ist hat der Fehler $P^{-1} \cdot e$ auch das Gewicht $d$.
- Wir decodieren $c'$ mit dem effizienten Decoder für $\Gamma$ und erhalten $S \cdot m$; es können bis zu $d$ Fehler effizient decodiert werden.
- Multiplikation mit $S^{-1}$ liefert den Klartext $m$.

### 5.5.4 Eigenschaften des Verschlüsselungsverfahrens

Aus dem Entschlüsselungs- / Decodier-Algorithmus ist klar, dass wieder die ursprÃŒngliche Nachricht gefunden wird (Korrektheit).
Bei Parametern ab $(1770, 1306, 43)$ wird das Verfahren aktuell als sicher betrachtet, die Sicherheit entspricht dann etwas der des RSA-Verfahrens mit einem Modulus von 1200 Bit.
Um die Schlüsselgröße von knapp 290 kByte zu reduzieren gibt es Ansätze die eine systematische Codierung verwenden. Dabei müß aber einiges beachtet werden, was wir hier nicht näher betrachten.

## 5.6 Decodierung des Walsh-Hadamard Codes

In Abschnitt 5.1 wurde der Walsch-Hadamard Code definiert und seine Parameter bestimmt. Der Walsh-Hadamard-Code $WH_k$ ist ein binärer $[2^k-1, k, 2^{k-1}]$-Code. Es können also $\lfloor \frac{d-1}{2} \rfloor = \lfloor \frac{2^k-2}{4} \rfloor = \lfloor \frac{n}{4} \rfloor$ Fehler eindeutig korrigiert werden. Diese hohe Korrekturrate ist bei Kanälen mit sehr viel Rauschen sinnvoll. Bei der Mariner 9 Mission der NASA wurde 1971 ein (punktierter) Walsh-Hadamard-Code (ein $[32, 6, 16]$-Code) eingesetzt um Bilder vom Mars zu übertragen. Er hat eine ähnliche Datenrate wie der 5-fache Wiederholungscode, kann aber Fehler besser korrigieren.
Uns interessieren hier allerdings Codes von größerer Länge um ein neues Decodierprinzip zu erläutern. Wir untersuchen hier die lokale Decodierung, das bedeutet, es werden nur wenige Codewortstellen „angefragt“ und für die Decodierung eines einzelnen Bits nicht das ganze Codewort betrachtet. Dies wird durch einen randomisierte Decoder realisiert, bei dem mit einer geringen Wahrscheinlichkeit einzelne Stellen auch falsch decodiert werden können.
Die Generatormatrix $G$ des WH-Codes enthält alle Vektoren $\neq 0$ der Länge $k$ über $\mathbb{F}_2$.
Sei $c \in WH_k = \langle m, x \rangle_{x \in \mathbb{F}_2^k \setminus \{0\}}$ ein fehlerfrei übertragenes Codewort, so bezeichnet $c_{e_i}$ mit $i = 1, \ldots, k$ also die Komponente von $c$, die durch Multiplikation mit $e_i$ in $G$ entsteht. Wegen $c_{e_i} = \langle m, e_i \rangle = m_i$, können die Komponenten $m_i$ von $m$ direkt gefunden werden (von der systematischen Codierung ist dieses Vorgehen bereits bekannt). Ist kein Fehler Aufgetreten, so kann das Nachrichtenwort also direkt aus dem Codewort an den entsprechenden Stellen abgelesen werden: $m = (c_{e_1}, \ldots, c_{e_k})$. Mit dieser Vorüberlegung können wir das folgende Lemma einfach beweisen.

**Lemma 25.** *Es sei $WH_k$ der Walsh-Hadamard-Code Länge $n = 2^k - 1$. Dann existiert ein randomisierter Decoder für $WH_k$, welcher bis zu $(\frac{1}{4} - \epsilon) \cdot n$ Fehler decodiert.*

**Beweis**. Sei $\tilde{c} = c + e$ das empfangene Wort, $c$ ein Codewort und $e$ der Fehlervektor vom Gewicht $< \left(\frac{1}{4} - \epsilon\right) \cdot n$. Es ist also höchstens ein $\left(\frac{1}{4} - \epsilon\right)$-Anteil von $\tilde{c}$ fehlerbehaftet.
Für jede Stelle ist die Fehlerwahrscheinlichkeit $< (\frac{1}{4} - \epsilon)$, also gilt für alle $x \in \mathbb{F}_2^k \setminus \{0\}$, dass $\langle m, x \rangle \neq \tilde{c}_x$ mit Wahrscheinlichkeit $< \left(\frac{1}{4} - \epsilon\right)$.
Sei nun $x \in \mathbb{F}_2^k$ zufällig gezogen, dann ist $wkt\{\tilde{c}_x \neq \langle m, x \rangle\} \leq \frac{1}{4} - \epsilon$ und ebenso auch für den „verschobenend Vektor“ $wkt\{\tilde{c}_{x+e_i} \neq \langle m, x + e_i \rangle\} \leq \frac{1}{4} - \epsilon$.

Fasst man dies beiden Ereignisse zusammen, so ergibt sich mit der folgenden Rechnung:

$$\begin{aligned} c_x + c_{x+e_i} &= \langle m, x\rangle + \langle m, x+e_i\rangle \\ &= \langle m, x\rangle + \langle m, x\rangle + \langle m, e_i\rangle \\ &= \langle m, e_i\rangle \\ &= m_i \end{aligned}$$

und $wkt\{\tilde{c}_x \neq c_x\} = wkt\{\tilde{c}_{x+e_i} \neq c_{x+e_i}\} \leq \frac{1}{4} - \epsilon$, dass

$$\begin{aligned} \tilde{c}_x + \tilde{c}_{x+e_i} &\neq m_i \quad \text{mit Wahrscheinlichkeit} \leq 2 \cdot \left(\frac{1}{4} - \epsilon\right) = \frac{1}{2} - 2 \cdot \epsilon \\ \Rightarrow \tilde{c}_x + \tilde{c}_{x+e_i} &= m_i \quad \text{mit Wahrscheinlichkeit} > \frac{1}{2} + 2 \cdot \epsilon \end{aligned}$$

Da die Erfolgswahrscheinlichkeit $> \frac{1}{2}$ kann durch $l$-faches Wiederholen und Mehrheitsentscheid die Fehlerwahrscheinlichkeit beliebig reduziert werden. Bei $l$-fachem Wiederholen ist die Fehlerwahrscheinlichkeit $\leq 2^{-\epsilon \cdot l}$ (Chernoff Schranke).

□

Das Vorgehen im Beweis ergibt den folgenden

**Decodieralgorithmus für das $i$-te Nachrichtenbit:**

1. Wähle $x$ zufällig,
2. berechne $\tilde{c}_x + \tilde{c}_{x+e_i}$ ($= m_i$ mit der Wahrscheinlichkeit $\geq \frac{1}{2} + 2 \cdot \epsilon$).
3. Wiederhole 1. und 2. genügend oft;
4. ein Mehrheitsentscheid liefert dann $m_i$.

## 5.7 Walsh-Hadamard-Code / List-Decodierung

Durch obigen Algorithmus können bis zu $\frac{n}{4} \approx \frac{d-1}{2}$ Fehler korrigiert werden. Das hört sich zunächst nach viel an, es stellt sich aber die Frage: „Wieviel (empfangene) Worte befinden sich denn innerhalb der Decodierkugeln?“
Das folgende Beispiel gibt ein ernüchterndes Bild.

**Beispiel.** Für den Walsh-Hadamard-Code $WH_k$ der Länge $n$ sind für ein paar kleine Werte von $k$ der Radius der Decodierkugeln und der prozentuale Anteil der Kugeln im gesamten Raum $\mathbb{F}_2^n$ angegeben. Auch außerhalb der Kugeln gibt es viele Worte $\tilde{c}$ bei denen ein einzelnes Codewort existiert, das $\tilde{c}$ am nächsten liegt und damit eindeutig decodierbar ist. Ist man beim Decodieren auch mit einer kurzen Liste von Codeworten (und ihrem Abstand zu $\tilde{c}$) zufrieden, so spricht man von List-Decoding. Der Prozentsatz an Worten, bei denen eine Liste mit bis zu 3 Codeworten mit demselben Abstand zu $\tilde{c}$ ausreicht ist in der Tabelle ebenfalls angegeben.

| k | n | Kugelradius | % Kugeln | eindeutig decodierbar | 2-3 Codeworte gleich weit |
|---|---|---|---|---|---|
| 4 | 15 | 3 | 28 % | 48 % | 80 % |
| 5 | 31 | 7 | 5,3 % | 52 % | 85 % |
| 6 | 63 | 15 | 0,1 % | 58 % | 90 % |
| 7 | 127 | 31 | 0,00004 % | 65 % | 94 % |

Diese Beispiel legt die Idee der List-Decodierung nahe. Zu einem empfangenen Wort wird eine Liste möglicher Codworte bzw. Nachrichtenworte ausgegeben die dem empfangenen Wort nahe sind.

**Satz 26.** ***(Goldreich-Levin)*** *Für jedes $\epsilon > 0$ existiert ein effizienter List-Decoder für $WH_k$, welcher $\left(\frac{1}{2} - \epsilon\right) \cdot n$ Fehler korrigiert und eine poly$(k)$-lange Liste an Kandidaten ausgibt.*

Diesen Satz werden wir nicht exakt beweisen. Wir geben hier den Algorithmus, der eine solche Liste liefert an und Begründungen weshalb er funktioniert:
Wir modifizieren den Algorithmus von vorher so, dass er auch noch bei mehr als $\frac{d-1}{2}$ Fehlern decodieren kann. In seiner ursprünglichen Form liefert er bei mehr als $\frac{d-1}{2}$ falschen Bits aber zufällige $m_i$ (mit Wahrscheinlichkeit $> \frac{1}{2}$ das falsche Bit) und ist so nicht zu verwenden.
Sei nun an $l$ Stellen $x_1, \ldots, x_l$ bekannt dass hier kein Fehler aufgetreten ist, also

$$\langle m, x_j \rangle = c_{x_j} = \tilde{c}_{x_j} \quad \text{für } j = 1, \ldots, l.$$

Für die anderen Stellen sei die Fehlerwahrscheinlichkeit $< \frac{1}{2} - \epsilon$. Dann kann $m_i$ mit dem ursprünglichen Decoder für $WH_k$ berechnet werden:

**Modifizierter Decodieralgorithmus für das $i$-te Nachrichtenbit (Version 1):**

1. Wähle $x$ zufällig aus $\{x_1, \ldots, x_l\}$,
2. berechne $\tilde{c}_x + \tilde{c}_{x+e_i} = c_x + \tilde{c}_{x+e_i}$ $(= m_i$ mit der Wahrscheinlichkeit $\geq \frac{1}{2} + \epsilon)$.
3. Wiederhole 1. und 2. genügend oft, z.B. für alle $x_1, \ldots, x_l$;
4. ein Mehrheitsentscheid liefert dann $m_i$.

Bei $l$ Wiederholungen ist die Fehlerwahrscheinlichkeit wieder $\leq \dfrac{1}{2^{\epsilon \cdot l}}$
Dieser leicht modifizierte Decodieralgorithmus funktioniert jetzt auch bei einer Fehlerrate bis zu $\frac{1}{2}$, die Bedingung ist aber, dass wir $l$ Stellen „sicher kennen“. Wie können wir dieses Problem angehen? Eine recht ungewöhnliche, aber sehr einfache Herangehensweise ist:

**Idee.** *Für die $l$ Stellen im empfangenen Wort $c_{x_1}, \ldots, c_{x_l} \in \mathbb{F}_2$ gibt es (nur) $2^l$ viele Möglichkeiten. Eine davon ist die richtige.*
*Probiere alle Möglichkeiten durch (die empfangenen Werte von $\tilde{c}_{x_i}$ werden nicht beachtet).*

**Modifizierter Decodieralgorithmus (Version 2):**

1. *Berechne für jede Belegung von* $\{c_{x_1}, \ldots, c_{x_l}\}$*:*
   - *berechne* $(m_1, \ldots, m_k)$ *mit dem Modifizierten Decodieralgorithmus (Version 1)*
   - *teste durch Codieren von* $(m_1, \ldots, m_k)$ *ob diese Nachricht möglich ist, d.h. die erwartete Fehler-Wahrscheinlichkeit hat.*
   - *speichere oder verwerfe* $(m_1, \ldots, m_k)$*.*
2. *Gebe die gespeicherte Liste von Nachrichtenwörtern aus.*

Dieser Algorithmus liefert auf jeden Fall die gesuchte Liste von Nachrichtenworten, der Aufwand ist aber mit $2^l$ Durchläufen des „Modifizierten Decodieralgorithmus (Version 1)“ zu groß.
Da es sich bei der Codierung um eine lineare Abbildung handelt lassen sich $l$ richtige Stellen $c_{x_1}, \ldots, c_{x_l} \in \mathbb{F}_2$ aber auch viel schneller finden:

- Setze $t := \lceil \log(l) \rceil$.
- Wähle $s_1, \ldots, s_t \in \mathbb{F}_2^k$ (wie $x_j$ vorher).
- Sind $\tilde{c}_{s_j} = c_{s_j}$ für $j = 1, \ldots, t$ fehlerfrei übertragen, dann ist auch $c_{s_1} \oplus c_{s_2} = \langle m, s_1 \rangle \oplus \langle m, s_2 \rangle = \langle m, s_1 \oplus s_2 \rangle$ fehlerfrei und damit sind alle $2^t$ vielen Stellen $\sum b_j c_{s_j} = c_{\sum b_j s_j}$ für $b_1, \ldots, b_t \in \mathbb{F}_2$ fehlerfrei.

Durch dieses Vorgehen erzeugen wir einen korrekten Satz von $l$ Stellen für das empfangene Wort indem wir $O(2^t) = O(l)$ viele Belegungen durchprobieren. Dies liefert einen effizienten List-Decoder für den Walsh-Hadamard-Code, der bis zu $\frac{n}{2}$ Fehler „korrigieren“ kann.

# 6 Konstruktion von Codes aus anderen Codes

## 6.1 Golay-Codes

Es gibt viele Varianten die Golay-Codes $\mathcal{G}_{24}$ und $\mathcal{G}_{23}$ zu konstruieren. Hier werden zwei verschiedene genannt. Der Code $\mathcal{G}_{24}$ ist ein $[24, 12]$-Code über $\mathbb{F}_2$ mit maximaler Minimaldistanz (die ist 8). Das ist auch die Idee für das erste Konstruktionsverfahren:

### 1. Variante der Konstruktion des $\mathcal{G}_{24}$

- Wähle ein Wort mit Gewicht 8
- Suche (z.B. durch lexikographisches Aufzählen) ein Wort, das zu den bisher erzeugten Abstand 8 hat.
- Endet mit $2^{12}$ Vektoren und liefert einen $[24, 12, 8]$-Code.

**2. Variante der Konstruktion des $\mathcal{G}_{24}$**

- Betrachte den Hamming-Code mit der Matrix

$$\mathcal{G}^t = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

$$\mathcal{G}'^t = \left(\begin{array}{ccc|c} & & & 1 \\ & \mathcal{G} & & 1 \\ & & & 0 \\ & & & 1 \end{array}\right)$$

  ist $[8, 4, 4]$-Code und erweitert den Hamming-Code.

- Definiere einen zu $\mathcal{G}$ äquivalenten Code

$$\overline{\mathcal{G}}^t = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

$$\overline{\mathcal{G}}'^t = \left(\begin{array}{ccc|c} & & & 1 \\ & \overline{\mathcal{G}} & & 1 \\ & & & 0 \\ & & & 1 \end{array}\right)$$

- $\mathcal{G}'$ und $\overline{\mathcal{G}}'$ generieren nur Codewörter mit Gewicht 0, 4 und 8.

  Der Hamming-Code hat Minimalgewicht 3, durch Ergänzen auf gerade Parität ergibt sich ein Code $\mathcal{G}'$ mit Codeworten von geradem Gewicht, also dem Minimalgewicht 4. Da $\mathbf{1} := (1, 1, 1, 1, 1, 1, 1, 1) \in \mathcal{G}'$ kann kein anderes Codewort mit Gewicht $> 4$ existieren, es hätte einen Minimalabstand $< 4$ von $\mathbf{1}$.

- Die beiden erweiterten Codes sind selbstdual, es gilt also $\mathcal{G}' = \mathcal{G}'^{\perp}$ und $\overline{\mathcal{G}}' = \overline{\mathcal{G}}'^{\perp}$

  Man kann einfach nachrechnen, dass $\mathcal{G}'^t \cdot \mathcal{G}' = 0$ und $k = n/2$ was die selbstdualtität von $\mathcal{G}'$ zeigt. Für $\overline{\mathcal{G}}'$ gilt dasselbe Argument.

- Die Schnittmenge der beiden erweiterten Codes ist trivial, d.h. $\mathcal{G}' \cap \overline{\mathcal{G}}' = \{0, 1\}$.

  Darstellen der Codeworte mit je 4 Unbestimmten und anschließender Koeffizientenvergleich liefert die Behauptung.

- Verwende $\mathcal{G}'$ und $\overline{\mathcal{G}}'$ um einen anderen Code zu definieren

$$\mathcal{G}_{24} := \{(a + x, b + x, a + b + x) | a, b \in \mathcal{G}', x \in \overline{\mathcal{G}}'\}$$

  Für die Generatormatrix bedeutet das:

$$\mathcal{G}_{24} := \begin{pmatrix} \mathcal{G}' & 0 & \overline{\mathcal{G}}' \\ 0 & \mathcal{G}' & \overline{\mathcal{G}}' \\ \mathcal{G}' & \mathcal{G}' & \overline{\mathcal{G}}' \end{pmatrix}_{(24\times 12)}$$

**Eigenschaften des $\mathcal{G}_{24}$**

- $\mathcal{G}_{24}$ ist ein selbstdualer $(24, 12)$-Code.

  Die Dimension kann man z. B. über $\text{rang}(\mathcal{G}_{24}) = 12$ berechnen. Die Selbstdualität ergibt sich wie oben durch $\mathcal{G}_{24}^t \cdot \mathcal{G}_{24} = 0$ und $k = n/2$.

- Die Gewichte der Codeworte von $\mathcal{G}_{24}$ sind durch 4 teilbar.

  Die Gewichte der 12 Basisvektoren sind durch 4 teilbar, es genügt also zu zeigen: $w(c) \equiv w(d) \equiv 0 \pmod 4 \Longrightarrow w(c+d) \equiv 0 \pmod 4$ für $c, d \in \mathcal{G}_{24}$.

  Definiere

  $$(c \cap d)_i = \begin{cases} 1 & : \quad c_i = d_i = 1 \\ 0 & : \quad \text{sonst} \end{cases}$$

  Dann gilt $\langle c, d \rangle \equiv w(c \cap d) \equiv 0 \pmod 2$, da $\mathcal{G}_{24}$ selbstdual ist. Das Gewicht $w(c + d) = w(c) + w(d) - 2 \cdot w(c \cap d)$ ist daher Summe/Differenz von durch 4 teilbaren Zahlen also auch durch 4 teilbar.

- $\mathcal{G}_{24}$ hat Minimalgewicht 8.

  Offensichtlich enthält $\mathcal{G}_{24}$ Worte vom Gewicht 8, es ist also nur noch zu zeigen, dass $\mathcal{G}_{24}$ kein Wort vom Gewicht 4 enthält.

  **Annahme:** $c = (a + x, b + x, a + b + x) \in \mathcal{G}_{24}$ hat Gewicht 4.

  In jeder der drei Komponenten ergänzt das 8-te Bit die anderen 7 auf gerade Parität, die drei Komponenten haben also gerades Gewicht und bei Gesamtgewicht 4 muss eine der Komponenten 0 sein.

  Ohne Beschränkung der Allgemeinheit sei $a + x = 0$ und damit $a = x \in \mathcal{G}' \cap \overline{\mathcal{G}}' = \{0, 1\}$.

  Im Fall $a = 0$ hat $c = (0, b, b)$ ein Gewicht von mindestens 8, im Fall $a = 1$ hat $c = (0, b + 1, b)$ ein Gewicht von mindestens 8.

  (Für einen ausführlicheren Beweis, vergleiche z.B. `http://www2.iwr.uni-heidelberg.de/groups/compalg/matzat/PDF/Codierungstheorie.pdf`, S.43)

- Es treten nur die Hamming-Gewichte 0, 8, 12, 16 und 24 auf.

- $\mathcal{G}_{24}$ ist $[24, 12, 8]$-Code.

**Konstruktion des $\mathcal{G}_{23}$**

- Streichen einer Spalte der Matrix $\mathcal{G}_{24}$, liefert $\mathcal{G}_{23}$

**Eigenschaften des $\mathcal{G}_{23}$**

- $\mathcal{G}_{23}$ ist ein $[23, 12, 7]$-Code.

- Es gilt $2^{23} = 2^{12} \cdot \left(1 + \binom{23}{1} + \binom{23}{2} + \binom{23}{3}\right) = 2^{12} \cdot \left(1 + 23 + 23 \cdot 22 + \frac{23 \cdot 22 \cdot 21}{2 \cdot 3}\right)$ und somit ist $\mathcal{G}_{23}$ perfekt.

## 6.2 Reed-Müller-Codes

**Satz 27.** ***(Die Summe von Codes)*** *Es seien zwei Codes $C_1$ als $[n, k_1, d_1]$-Code und $C_2$ als $[n, k_2, d_2]$-Code gegeben. Betrachte nun den Code $\{(a_1, a_1 + a_2) | a_1 \in C_1, a_2 \in C_2\}$. Dies ist ein $[2n, k_1 + k_2, d_{min}]$-Code mit Generatormatrix*

$$\mathcal{G}_1 \ \& \ \mathcal{G}_2 := \begin{pmatrix} \mathcal{G}_1 & 0 \\ \mathcal{G}_1 & \mathcal{G}_2 \end{pmatrix}.$$

*Es gilt $d_{min} = \min\{2d_1, d_2\}$.*

**Beweis**.

- Offensichtlich ist $d_{min} \leq \min\{2d_1, d_2\}$ es bleibt also zu zeigen: $d_{min} \geq \min\{2d_1, d_2\}$
- Fall 1: $a_2 = 0 \Rightarrow w(a_1, a_1) \geq 2d_1$
- Fall 2: $a_2 \neq 0 \Rightarrow w(a_1, a_1 + a_2) = w(a_1) + w(a_1 + a_2) = w(-a_1) + w(a_1 + a_2) \geq w(-a_1 + a_1 + a_2) = w(a_2) \geq d_2$

□

**Definition 18. (Reed-Muller-Code)** Sei $r, m \in \mathbb{N}, 0 \leq r \leq m$. $RM(r, m)$ ist ein $[n, k, d]$-Code über $\mathbb{F}_2$ mit $n = 2^m$, $k = \sum_{i=0}^{r} \binom{m}{i}$, $d = 2^{m-r}$. Er wird rekursiv definiert durch:

- $RM(0, m)$ ist ein $[2^m, 1, 2^m]$-Wiederholungscode.
- $RM(m, m)$ ist uncodiert, das heißt, das Informationswort wird direkt ausgegeben.
- $RM(r + 1, m + 1) := RM(r + 1, m) \ \& \ RM(r, m)$

**Beispiel**

$$RM(1, 1) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, d_{min} = 1$$

$$RM(1, 2) = RM(1, 1) \ \& \ RM(0, 1) = \left(\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \hline 1 & 0 & 1 \\ 0 & 1 & 1 \end{array}\right), d_{min} = 2$$

$$RM(1, 3) = RM(1, 2) \ \& \ RM(0, 2) = \left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{array}\right), d_{min} = 4$$

$$RM(2,2) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, d_{min} = 1$$

$$RM(2,3) = RM(2,2) \ \& \ RM(1,2) = \left(\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right), d_{min} = 2$$

## 6.3 Verkettete Codes (Concatenated Codes)

Vereinfacht sagt die Gilbert-Varshamov-Schranke „für eine feste Rate $R \leq 1 - H(\delta) - \epsilon$ existiert ein $[n, R \cdot n, \delta \cdot n]$-Code“ (dabei nennen wir $\delta$ die relative Minimaldistanz). Einen solchen Code zu „finden“ ist aber nicht einfach, zudem wäre ein solcher „zufälliger“ Code auch nicht effizient decodierbar. Allgemein benötigt man einen Aufwand in der Größenordnung $2^{\Omega(n)}$ um einen zufälligen Code zu decodieren.
Bisher haben wir einige Familien von algebraischen Codes betrachtet, die auch effizient decodierbar sind. Sie haben aber alle einen Nachteil wenn man das Verhalten bei wachsendem $n$ betrachtet:

1. Hamming-Codes:
   Sie haben eine feste Minimaldistanz $d = 3$. Daher konvergiert für $n \to \infty$ die relative Minimaldistanz $\delta = \frac{d}{n}$ gegen 0.

2. Walsh-Hadamard-Codes:
   Die Rate ist $R = \frac{k}{n} = \frac{k}{2^k - 1}$, sie konvergiert für $n \to \infty$ gegen 0.

3. Reed Solomon Codes:
   Mit wachsendem $n$ muss das Alphabet des Grundkörpers $q$ vergrössert werden, so dass in $\mathbb{F}_q$ mindestens $n$ Auswertugsstellen zur Verfügung stehen.

4. andere Codes:
   Bei den anderen betrachteten Codes ist die Situation nicht besser, das ist allerdings nicht so einfach zu sehen.

### 6.3.1 Allgemeine Konstruktion

Ziel dieses Abschnitts ist es eine Familie von Codes zu finden, bei der sowohl die Rate $R$, als auch die relative Minimaldistanz $\delta$ für $n \to \infty$ beide nicht gegen 0 gehen, also größer als $\epsilon_1, \epsilon_2 > 0$ bleiben.

**Definition 19.** Eine Familie von Codes heißt asymptotisch gut, falls für alle $n \in \mathbb{N}$ sowohl für die Rate $R \geq \epsilon_1 > 0$, also auch die relative Minimaldistanz $\delta \geq \epsilon_2 > 0$ gilt (für geeignete $\epsilon_1, \epsilon_2$).

Es wurde lange vermutet, dass asymptotisch gute Codes, die effizient decodierbar sind, nicht existieren. Die Idee der verketten Codes brachte 1972 dann eine Lösung: Es wird ein Wort codiert und das entstandene Codewort mit einem anderen Code nochmals codiert. Dadurch werden Codes mir verschiedenen Eigenschaften kombiniert um mehrere benötigte Eigenschaften gleichzeitig zu erreichen.
Die beiden Codes seien:

- $C_{in}$ ein $[n_1, k_1, d_1]$-Code über $F_2$ (innerer Code)
- $C_{out}$ ein $[n_2, k_2, d_2]$-Code über $\mathbb{F}_q$ mit $q = 2^{k_1}$ (äußerer Code)

Codiert und decodiert wird dann entsprechend Abb. 1, die Hintereinanderausführung $C_{out} \circ C_{in}$ heißt dann verketteter Code.

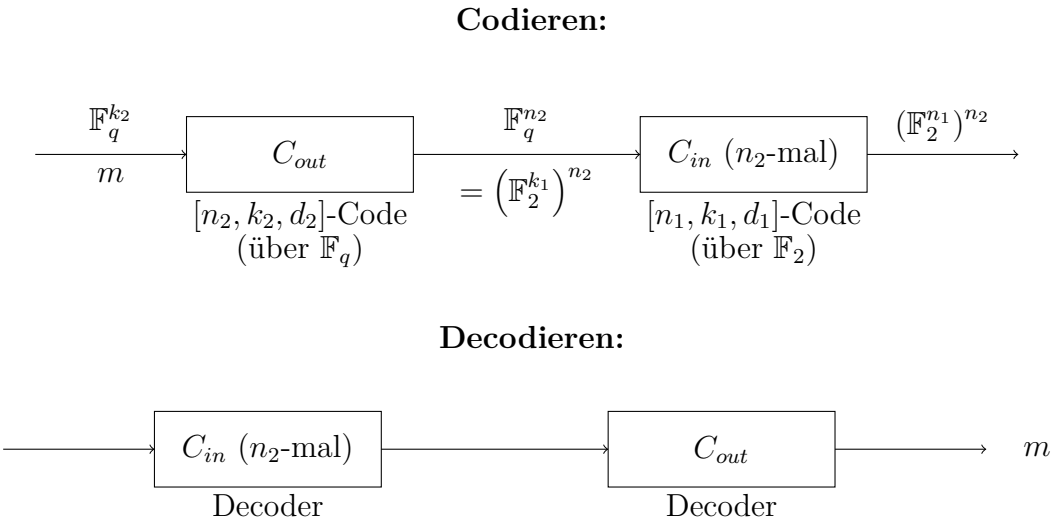


Abbildung 1: Codieren und Decodieren eines verketteten Codes

**Lemma 28.** *Es sei $C_{in}$ ein $[n_1, k_1, d_1]$-Code über $\mathbb{F}_2$, $C_{out}$ ein $[n_2, k_2, d_2]$-Code über $\mathbb{F}_q$ mit $q = 2^{k_1}$, dann ist $C_{out} \circ C_{in}$ (entsprechend Add. 1) ein binärer $[n_1 \cdot n_2, k_1 \cdot k_2, d]$-Code mit $d \geq d_1 \cdot d_2$.*

**Beweis**. Es muss für alle drei Parameter gezeigt werden dass sie den Angaben genügen. Dass ein binärer Code entsteht ist offensichtlich, bei $n$ und $k$ genügt „schafes Hinsehen“:

- **Länge:** Es entstehen $n_2$ viele Blöcke der Länge $n_1$, also ist die Länge $n_1 \cdot n_2$
- **Dimension:** Es gilt analog, dass die Dimension $k_1 \cdot k_2$ ist.

- **Minimaldistanz:** Sei $0 \neq m \in \mathbb{F}_q^{k_2}$ eine Nachrich. Dann hat $C_{out}(m) \in \mathbb{F}_q^{n_2}$ mindestens $d_2$ viele Komponenten ($\in F_q$) die ungleich 0 sind. Jede dieser $d_2$ Komponenten wird als binäres Wort ($\in \mathbb{F}_2^{k_1}$) aufgefasst und liefert durch Codieren mit $C_{in}$ einen Bitvektor mit mindestens $d_1$ Bit ungleich 0. Insgesamt sind also mindestens $d_1 \cdot d_2$ Bit ungleich 0. □

Dass die verketteten Codes bei geeigneter Wahl von $C_{in}$ und $C_{out}$ asymptotisch gute Codes liefern betrachten wir später, zunächst betrachten wir das Decodieren:

**Lemma 29.** *Existiert ein (effizienter) Decoder für $C_{in}$, der $t_1$ Fehler korrigiert und ein (effizienter) Decoder für $C_{out}$, der $t_2$ Fehler korrigiert, dann liefert die „Kombination" einen (effizienten) Decoder, der $t_1 \cdot t_2$ Fehler korrigiert.*

**Beweis**. Sei $\overline{c} = c+e \in \mathbb{F}_2^{n_1 \cdot n_2}$ wobei $c$ ein Codewort von $C_{out} \circ C_{in}$ ist und der Fehlervektor $e$ maximal das Gewicht $t_1 \cdot t_2$ hat.
Von den $n_2$ Blöcken der Länge $n_1$ werden nur diejenigen vom „$C_{in}$-Decoder" falsch decodiert, die mehr als $t_1$ Fehler enthalten. Davon existieren maximal $t_2$ viele. Nach dieser inneren Decodierung entsteht ein Wort aus $F_q^{n_2}$ mit maximal $t_2$ vielen Fehlern, diese korrigiert der Decoder von $C_{out}$. □

### 6.3.2 Konkrete Instanziierung

Aus dem Lemma von eben ergibt sich die recht einfache

**Beobachtung 30.** *Sind sowohl $C_{out}$, als auch $C_{in}$ asymptotisch gut, so auch $C_{out} \circ C_{in}$.*

**Beweis**. Sei $C_{out}$ ein $[n_2, R_2 \cdot n_2, \delta_2 \cdot n_2]$-Code und $C_{in}$ ein $[n_1, R_1 \cdot n_1, \delta_1 \cdot n_1]$-Code, dann ist $C_{out} \circ C_{in}$ ein $[n_1 n_2, R_1 R_2 \cdot n_1 n_2, \delta_1 \delta_2 \cdot n_1 n_2]$-Code und damit auch asymptotisch gut. □

Da wir am Anfang des Abschnittes ja erwähnt habe, dass man keine asymptotisch guten Codes kannte (die auch effizient decodierbar sind) erscheint diese Beobachtung auf den ersten Blick unsinnig.
Am Anfang des Abschnittes haben wir aber auch zwei weitere Eigenschaften gesehen, mit denen diese Beobachtung helfen kann einen asymptotisch guten Code zu konstruieren: Es existieren asymptotisch gute Codes (wegen der Gilbert-Varshamov-Schranke), sie sind allerdings nicht einfach zu finden und nicht einfach zu decodieren. Sind solche Codes sehr kurz, so kann eine vollständige Suche nach den Codes aber machbar sein - auch das Decodieren ist bei sehr kurzen Codes machbar. Reed Solomon Codes sind in einem gewissen Sinn asymptotisch gut: Wählt man $k \approx n/2$, dann ist die Rate des Codes etwa $1/2$, die Minimaldistanz ist dann $n-k+1 \approx 1/2$ und damit ist die relative Minimaldistanz auch etwa $1/2$. Das Problem bei dieser Konstruktion ist, dass das zugrunde liegende Alphabet bei wachsendem $n$ auch wachsen muss.
Verwenden wir einen RS-Code aus äußeren Code und fangen die wachsende Grösse des Grundkörpers mit einem inneren Code (über $\mathbb{F}_2$) auf, so können wir mit Beobachtung 30 einen asymptotisch guten Code konstruieren.
Mit dieser Idee müssen nur noch die verschiedenen Parameter richtig gewählt werden:

- Sei $n$ die Länge des (verketteten) Codes.

- Setze $n_1 :\approx \frac{\log(n)}{R_1}$ und $n_2 :\approx R_1 \frac{n}{\log(n)}$
- Geeignete Rundungen liefern dann $n = n_1 \cdot n_2$.
- Es ergibt sich: $q = 2^{k_1} = 2^{R_1 \cdot n_1} \approx 2^{\log(n)} = n > n_2$, womit wir $C_{out}$ als RS-Code über $\mathbb{F}_q$ definieren können, damit ist $C_{out}$ effizient decodierbar.
- Wir benötinge jetzt noch einen effizient decodierbaren inneren Code $C_{in}$.
- $C_{in}$ muss ein binärer $\left[\frac{\log(n)}{R_1}, \log(n), \frac{\delta_1}{R_1}\log(n)\right]$-Code sein.
- Damit hat $C_{in}$ aber nur $2^{k_1} = 2^{\log(n)} = n$ viele Codeworte und kann über vollständige Suche in $n$ Schritten decodiert werden. Dies ist effizient (linear in der Länge des (verketteten) Codes) machbar.
- Wir wählen daher einen zufälligen binären $[n_1, R_1 \cdot n_1]$-Code als inneren Code, $\delta_1$ lässt sich, wegen der geringen Länge, wieder durch vollständige Suche (effizient) berechnen. Wir testen dann ob $R_1 \geq 1 - H_1(\delta)$, wenn ja ist ein geeigneter innerer Code gefunden, ansonsten suchen wir weiter; ein geeigneter Code existiert nach der Gilbert-Varshamov-Schranke.

Damit ist eine Familie von effizient decodierbaren, asymptotisch guten Codes gefunden.

# A Endliche Körper

Ein Körper ist eine algebraische Struktur in der Addition, Subtraktion, Multiplikation und Division auf eine bestimmte Weise durchgeführt werden können.

**Definition 20.** Ein *Körper* ist eine Menge $K$ mit zwei inneren zweistelligen Verknüpfungen (Addition und Multiplikation), für die folgende Bedingungen erfüllt sind:

1. $(K, +, \cdot)$ mit $(K, +)$ und $(K \setminus \{0\}, \cdot)$ sind abelsche Gruppen mit 0 bzw. 1 als Neutralem Element.
2. Das additiv Inverse von $a \in K$ ist $-a$ (das negative von $a$). Das multiplikative Inverse von $a$ ist $a^{-1}$ (der Kehrwert von $a$).
3. Es gelten die Distributivgesetze.

Einige Eigenschaften von Körpern:

1. Ein Körper ist ein Ring.
2. Ein Körper $K$ ist nullteilerfrei, d.h. es gilt: $a \cdot b = 0 \Rightarrow a = 0$ oder $b = 0$.

**Definition 21.** Ein *endlicher Körper* ist ein Körper mit endlich vielen Elementen.

**Satz 31.** *Jeder endliche Körper hat $q = p^n$ Elemente wobei $p$ eine Primzahl ist und $n$ eine positiven natürlichen Zahl. Bis auf Isomorphie gibt es zu jedem $q = p^n$ genau einen endlichen Körper; er wird mit $\mathbb{F}_q$ bezeichnet.*

Endliche Körper werden häufig als Restklassenkörper dargestellt.
Ist $p$ eine Primzahl, so ist der Restklassenring $\mathbb{Z}/p\mathbb{Z}$ ein endlicher Körper mit $p$ Elementen.
Die anderen endlichen Körper kann man als Restklassen des Polynomrings $\mathbb{F}_p[x]$ konstruieren: Sei $f(x) \in \mathbb{F}_p[x]$ ein irreduzibles Polynom vom Grad $n$, dann ist $\mathbb{F}_p[x]/(f(x))$ ein Körper mit $q = p^n$ Elementen.

**Beispiel.** ${}^{\mathbb{F}_2[x]}\!/_{x^3+x+1}$ *ist ein Körper mit 8 Elementen, die* 8 *Polynome vom Grad* $< 3$ *sind ein Repräsentantensystem der Restklassen.*

**Beispiel.** ${}^{\mathbb{F}_2[y]}\!/_{y^3+y^2+1}$ *ist ebenfalls ein Körper mit 8 Elementen, die beiden Körper sind also isomorph.*
*Die Abbildung $x \mapsto y$ liefert allerdings keinen Isomorphismus:*

$$\begin{array}{rccl} \varphi: & {}^{\mathbb{F}_2[x]}\!/_{x^3+x+1} & \to & {}^{\mathbb{F}_2[y]}\!/_{y^3+y^2+1} \\ & x & \mapsto & y \\ \Rightarrow & x^2 & \mapsto & y^2 \\ \Rightarrow & (x+1) = x^3 & \mapsto & \varphi(x) \cdot \varphi(x^2) = y \cdot y^2 = y^3 = y^2 + 1 \\ \textit{aber} & x + 1 & \mapsto & \varphi(x) + \varphi(1) = y + 1 \end{array}$$

*Die Abbildung $x \mapsto y + 1$ liefert einen Isomorphismus — weshalb?*

# B Polynome über endlichen Körpern

Polynome werden auch ÃŒber endlichen Körpern wie gewohnt geschrieben.

$$\mathbb{F}_q[x] = \sum_{i=0}^{r} a_i \cdot x^i \text{ mit } a_i \in \mathbb{F}_q$$

$Grad(f(x))$ wie gewohnt
$Grad(f(x) + g(x)) \leq max\{Grad(f(x)) + Grad(g(x))\}$
$Grad(f(x) \cdot g(x)) = Grad(f(x)) + Grad(g(x))$
$\mathbb{F}_q[x]$ ist euklidischer Ring, es existieren "Division mit Rest".

$$f(x) = \underbrace{a(x)}_{Vielfaches} \cdot g(x) + \underbrace{r(x)}_{Rest} \text{ mit } Grad(r(x)) < Grad(g(x))$$

Auswerten von Polynomen ist Homomorphismus.
d.h. $f(b) \cdot g(b) = (f \cdot g)(b)$
analog für die Addition.

## B.1 Nullstellen von Polynomen

d.h. $f(a) = 0$
Betrachte $f(x)$ und $(x - a) \in \mathbb{F}_q[x]$ dann gilt

$$f(x) = (x - a) \cdot s(x) + r(x) \text{ mit } Grad(r(x)) < 1$$

$\Rightarrow f(a) = 0 = 0 \cdot s(x) + \underbrace{r(x)}_{=0}$
$Grad(f(x)) = k$
Es sei $a$ eine Nullstelle von $f(x)$
$\Rightarrow f(x) = g(x)(x - a)$
mit $Grad(g(x)) = k - 1$
Dies lässt sich maximal $k$ mal durchführen, das heißt $f(x)$ hat maximal $k$ Nullstellen.

# Literatur

[1] Thomas M. Cover und Joy A. Thomas; *Elements of Information Theory*; Wiley, 2006.

[2] J.H. van Lint; *Introduction to Coding Theory*; Springer, 1999.

[3] Madhu Sudan; *Skript/Folien zur Vorlesung: Algorithmic Introduction to Coding Theory*; http://people.csail.mit.edu/madhu/FT01/