

Grundlagen der Informatik

Prof. Dr. J. Schmidt

Fakultät für Informatik

GDI – WS 2020/21

Codesicherung und Kanalcodierung
Hamming, CRC, Reed-Solomon

- Wie können nicht-binäre Codes gesichert werden?
- Was ist der Hamming-Code und wie wird dieser gebildet?
- Was sind CRC-Codes?
- Wie funktionieren QR-Codes?



Sicherung nicht-binärer Codes (1)

Kapitel 4.2/4.3/4.4: Codesicherung und Kanalcodierung – Hamming, CRC, Reed-Solomon

- Code-Sicherung ist nicht nur auf binäre Codes beschränkt
- Manche Fehler in der Übertragung von z.B. **natürlicher Sprache** lassen sich aufgrund der Redundanz erkennen und korrigieren
 - d.h. die Korrektur ist aus dem Zusammenhang des Textes ersichtlich
- Manche Fehler führen zu gültigen Worten
 - d.h. sind nicht als Fehler erkennbar
 - (Zweideutigkeit)



Beispiele

- Textverarbeitungsprogramme
 - Fehlerkorrektur nach gültigen Rechtschreibregeln
- Natürliche Sprache
 - Erkennbare und ggf. korrigierbare Fehler in der natürlichen Sprache

Empfangener Text	Korrigierter Text	
Vorlesunk	Vorlesung	→ eindeutig korrigierbar
Vorlosung	Verlosung / Vorlesung ?	→ zweideutig
Der Memsch denkt	Der Mensch denkt	→ eindeutig korrigierbar
Der Mensch lenkt	Der Mensch denkt / lenkt ?	→ nicht erkennbar



Gegen Fehleingabe gesicherte Ziffern-Codes

- Bei Eingabe von Dezimalziffern (z.B. Bestellnummern) ist mit Fehlern zu rechnen
- Die meisten Methoden zur Aufdeckung von Fehleingaben arbeiten mit Prüfziffern

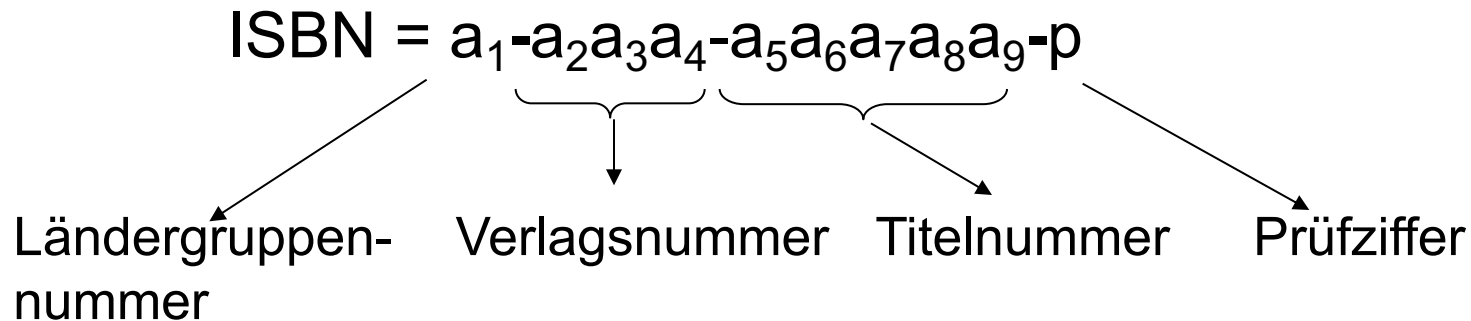


- Einfachste Methode: Bildung der **Quersumme**
 - Zur Reduktion auf eine Dezimalstelle → Quersumme Modulo 10
 - Divisionsrest stellt Prüfziffer dar, die am Ende der Zeichenreihe angefügt wird
- Fehlererkennung
 - Eingabe einer falschen Ziffer wird durch Vergleich der
 - resultierende Prüfziffer
 - mit der erwarteten Prüfziffererkannt
- Vertauschungsfehler
 - Vertauschen zweier Ziffern wird nicht erkannt
 - Lösungsansatz
 - Gewichtung der Ziffern bei der Quersummenbildung



Beispiel gesicherter Ziffern-Code

- 10-stellige ISBN-Buchnummer



- Berechnung von p

- $10a_1 + 9a_2 + 8a_3 + 7a_4 + 6a_5 + 5a_6 + 4a_7 + 3a_8 + 2a_9$
- Bestimmung von p so, dass die Summe durch die Addition mit p zu einer ohne Rest durch 11 teilbaren Zahl ergänzt wird



Beispiel gesicherter Ziffern-Code

- Daraus folgt: $0 \leq p \leq 10$
- Zweistellige Prüfziffer 10 wird durch Einzelzeichen X ersetzt
- Für eine korrekte ISBN-Nummer gilt:
$$(10a_1 + 9a_2 + 8a_3 + 7a_4 + 6a_5 + 5a_6 + 4a_7 + 3a_8 + 2a_9 + p) \bmod 11 = 0$$
- Sowohl falsch eingegebenen Ziffern als auch vertauschte Ziffern können erkannt werden



Aufgabe: Überprüfen Sie folgende ISBN
3-528-25717-2

- Ist sie korrekt oder ist bei der Eingabe ein Fehler aufgetreten?



Internationale Bankkontonummer (IBAN)

- bis zu 34 Zeichen, normalerweise kürzer
- in Deutschland: 22 Stellen:

DE $p_1 p_2 b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9 k_{10}$

- $k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9 k_{10}$: ehemalige Kontonummer
- $b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8$: ehemalige Bankleitzahl
- $p_1 p_2$: Prüfziffern



IBAN – Berechnung der Prüfziffer

- Initialisiere $p_1 p_2 = 00$
- stelle Länderkürzel und $p_1 p_2$ ganz nach rechts:

$b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9 k_{10}$ **DE 00**

- ersetze Länderkennung durch
Position des Buchstabens im Alphabet + 9 (A=10, B=11, ...):

$b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9 k_{10}$ **13 14 00**

- Berechne Rest bei Division durch 97
- Lege $p_1 p_2$ so fest, dass sich für den Rest 1 ergibt
- Beachte
 - ganzzahlige Arithmetik mit bis zu 36-stelligen Zahlen nötig
 - lässt sich mit Standarddatentypen in gängigen Programmiersprachen von bis zu 64 Bit nicht durchführen



IBAN – Beispiel: **Generierung**

BLZ: 711 500 00, Kontonummer: 215 632

- Kombination, Länderkürzel/Prüfziffern rechts:

711500000000215632**DE00**

- Ersetze Kürzel DE durch Positionen im Alphabet + 9:

711500000000215632**131400**

- Rest bei Division durch 97 liefert:

711500000000215632131400 mod 97 = 49

- Prüfziffern lauten also: $98 - 49 = 49$

- IBAN: DE 49 7115 0000 0000 2156 32



IBAN – Beispiel: **Validierung**

IBAN: DE 49 7115 0000 0000 2156 32

- Länderkürzel/Prüfziffern rechts:

711500000000215632**DE49**

- Ersetze Kürzel DE durch
Positionen im Alphabet + 9:

711500000000215632**131449**

- Rest bei Division durch 97 liefert:

711500000000215632131449 mod 97 = 1
→ IBAN korrekt



Berechnung aus $z_n \dots z_i \dots z_0$ (enthält Prüfziffern) mit Gewichten g_i

$$\sum_{i=0}^n g_i z_i \mod m = 0$$

- Erkennung **einzelner falscher Ziffern** ist garantiert, wenn alle Gewichte g_i teilerfremd zu m sind (d.h. $\text{ggT}(g_i, m) = 1$)
- Erkennung der **Vertauschung** zweier Ziffern z_i und z_k ist garantiert, wenn $g_i - g_k$ teilerfremd zu m ist.

→ Primzahlen als Modul m sinnvoll



Hamming-Code (1)

Kapitel 4.2/4.3/4.4: Codesicherung und Kanalcodierung – Hamming, CRC, Reed-Solomon

- Von R. Hamming entwickelt (1950)
 - 1-korrigierender Code
 - mit einer Hamming-Distanz von 3
- Idee
 - Einführung von Prüf-Bits
 - Deren binäre Codierung gibt an
 - > 0 → die Position des fehlerhaften Bits
 - $= 0$ → fehlerfreie Übertragung
 - Beispiel
 - 3 zusätzliche Prüf-Bits erlauben $2^3 = 8$ Zustände
 - → 7 Fehlerpositionen codierbar



- Einfachster Hamming-Code
 - (7,4)-Code
 - Block-Code der Länge 7, wobei
 - 4 Bits Nutzinformationen
 - 3 Bits zur Fehlerkorrektur
- Allgemein
 - Hamming-Codes der Länge $2^r - 1$, wobei $r \geq 3$ sein muss
 - r Paritätsbits (daraus ergeben sich dann Prüfbits)
 - $2^r - 1 - r$ Informations-Bits

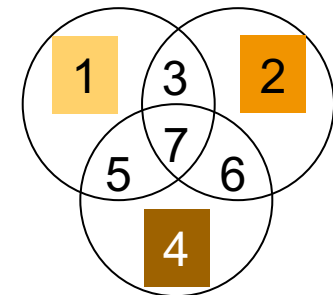


Grundsätzlicher Aufbau

- Paritätsbit-Positionen sind Zweierpotenzen
($1 = 2^0$, $2 = 2^1$, $4 = 2^2$, ...)
 - gerade Parität, Einsen
- Beispiel (7,4)-Code

7	6	5	4	3	2	1	Paritätsbit an Position
D	D	D	P	D	P	P	
D	-	D	-	D	-	P	2^0
D	D	-	-	D	P	-	2^1
D	D	D	P	-	-	-	2^2

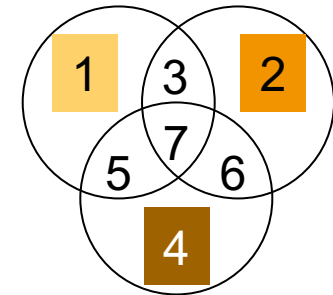
D ... Datenbit
P ... Paritäts-Bit



Hamming-Code (4)

Kapitel 4.2/4.3/4.4: Codesicherung und Kanalcodierung – Hamming, CRC, Reed-Solomon

Dezimal	D	D	D	P	D	P	P
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	0	1	1	0	0	1
3	0	0	1	1	1	1	0
4	0	1	0	1	0	1	0
5	0	1	0	1	1	0	1
6	0	1	1	0	0	1	1
7	0	1	1	0	1	0	0
8	1	0	0	1	0	1	1
9	1	0	0	1	1	0	0
10	1	0	1	0	0	1	0
11	1	0	1	0	1	0	1
12	1	1	0	0	0	0	1
13	1	1	0	0	1	1	0
14	1	1	1	1	0	0	0
15	1	1	1	1	1	1	1



- Hamming-Codes sind **optimal**
 - Jedes mögliche Wort ist entweder tatsächlich ein Codewort
 - oder hat eine Stellendistanz von 1 zu einem tatsächlichem Codewort



Hamming-Code (5)

Beispiel: Information 1101 ist zu übertragen
→ Bitfolge 1100110 wird übertragen

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

Bei Änderung eines der Bits 1 bis 7
→ eines oder mehrere der drei Paritäts-Bits sind betroffen

- Ändert man 7. Bit → Auswirkungen auf alle drei Paritäts-Bits
- Fehler beim 6. Bit → Auswirkung nur auf Paritäts-Bits 2 und 4
- Kippen eines Paritäts-Bits → Auswirkung nur auf gekipptes Bit



Hamming-Code – Beispiel

Kapitel 4.2/4.3/4.4: Codesicherung und Kanalcodierung – Hamming, CRC, Reed-Solomon

- Bitfolge 1100110 ist zu übertragen, wobei das 6. Bit kippt (Empfänger: 1000110)

7	6	5	4	3	2	1	Paritätsbit	Prüfbit wird gewertet als
1	0	0	0	1	1	0		
1	-	0	-	1	-	0	richtig	0
1	0	-	-	1	1	-	falsch	1
1	0	0	0	-	-	-	falsch	1

- Bitkombination aus letzter Spalte „von unten nach oben“
→ Dualzahl 110 (Dezimal 6)
→ im 6. Bit ist Fehler aufgetreten
- Die Prüfbits – dual codiert – geben die Position des fehlerhaften Bits an



Hamming-Code – (15, 11)

Kapitel 4.2/4.3/4.4: Codesicherung und Kanalcodierung – Hamming, CRC, Reed-Solomon

Aufbau des (15, 11) Hamming-Codes mit 4 Prüf-Bits

15	14	13	12	11	10	9	2^3 8	7	6	5	2^2 4	3	2^1 2	2^0 1
D	D	D	D	D	D	D	P	D	D	D	P	D	P	P
D	-	D	-	D	-	D	-	D	-	D	-	D	-	P
D	D	-	-	D	D	-	-	D	D	-	-	D	P	-
D	D	D	D	-	-	-	-	D	D	D	P	-	-	-
D	D	D	D	D	D	D	P	-	-	-	-	-	-	-

D ... Datenbit
P ... Paritäts-Bit



- Sie empfangen folgendes (7,4)-Hamming-Codewort:

1000111

- War die Übertragung fehlerfrei?
- Falls nein:
 - korrigieren Sie das Codewort!
 - welche Zahl wurde übertragen?



Cyclic Redundancy Check (CRC)

Kapitel 4.2/4.3/4.4: Codesicherung und Kanalcodierung – Hamming, CRC, Reed-Solomon

● Ziele:

- Fehlererkennung durch Hinzunahme von möglichst wenig Redundanz
- Erkennung von
 - Einzel- und Doppelfehlern
 - Burstfehlern (mehrere fehlerhafte Bit am Stück)
- einfache Implementierung (vor allem auch in Hardware)

● Verwendung z.B.

- Ethernet, USB, Bluetooth, SCSI, Serial ATA, ISDN, DECT (schnurlose Telefone), CAN, FlexRay (Automotive)
- ...



- hänge an eine n Bit lange Nachricht k Bit CRC-Code an
- fasse Nachricht als *Koeffizienten eines dyadischen Polynoms* auf
 - dyadisch = rechne modulo 2
 - Koeffizienten können also nur Werte 0 und 1 annehmen
- Beispiel
 - Nachricht: 10011010
 - Polynom $N(x)$
$$1 \cdot x^7 + 0 \cdot x^6 + 0 \cdot x^5 + 1 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x + 0 = x^7 + x^4 + x^3 + x$$



- wähle ein Polynom $C(x)$ vom Grad k
(k = Länge CRC-Code)
- übertrage ein Polynom $S(x)$, das ohne Rest durch $C(x)$ teilbar ist
- Beispiel $k = 3$
 - $C(x) = x^3 + x^2 + 1$
 - übertrage: $S(x) = N(x) + k \text{ Bit}$



● Schritte

- $T(x) = N(x) \cdot x^k$ (\rightarrow hänge k Nullen an Nachricht an)
- berechne Rest $R(x)$ bei Division $T(x) / C(x) \rightarrow T(x) \bmod C(x)$
- sende $S(x) = T(x) - R(x)$
 - bei mod 2 $\rightarrow T(x) - R(x) = T(x) + R(x)$
 - also: hänge $R(x)$ an $N(x)$ an

● Beispiel

- $N(x) = 10011010 = x^7 + x^4 + x^3 + x$
- $C(x) = 1101 = x^3 + x^2 + 1 \rightarrow k = 3$
- $T(x) = 10011010000 = x^{10} + x^7 + x^6 + x^4$
- $R(x) = 101 = x^2 + 1$
- $S(x) = 10011010101 = x^{10} + x^7 + x^6 + x^4 + x^2 + 1$



- alle Rechnungen mod 2
- daher gilt $1 + 1 = 1 - 1 = 0$
- Subtraktion kann durch **stellenweises XOR** erfolgen
- beginne immer beim linkensten Koeffizienten der Nachricht $N(x)$ (bzw. der Erweiterung $T(x)$)



CRC – Polynomdivision – Beispiel (Sender)

Kapitel 4.2/4.3/4.4: Codesicherung und Kanalcodierung – Hamming, CRC, Reed-Solomon

$$\begin{array}{llll} C(x) = & x^3 + x^2 + 1 & = 1101 & \text{Generator} \\ N(x) = & x^7 + x^4 + x^3 + x & = 10011010 & \text{Nachricht} \end{array}$$

Generator \swarrow

$$\begin{array}{r} 10011010000 \leftarrow T(x) = \text{Nachricht mit k Nullen} \\ \underline{1101} \quad \text{XOR} \\ 1001 \\ \underline{1101} \\ 1000 \\ \underline{1101} \\ 1011 \\ \underline{1101} \\ 1100 \\ \underline{1101} \\ 1000 \\ \underline{1101} \\ 101 \end{array}$$

\swarrow Rest $T(x) \bmod C(x)$

Übertrage Nachricht und
angehängten Rest:

$$S(x) = 10011010101$$



Schritte

- Empfange Polynom $S'(x)$
- berechne Rest $R'(x)$ bei Division
 $S'(x) / C(x) \rightarrow S'(x) \bmod C(x)$
 - Rest = 0
 - fehlerfreie Übertragung
 - oder nicht detektierbarer Fehler
 - Rest $\neq 0$
 - mindestens 1 Bit in Nachricht ist falsch
 - Nachricht muss nochmal gesendet werden



CRC – Beispiel (Empfänger, fehlerfrei)

Kapitel 4.2/4.3/4.4: Codesicherung und Kanalcodierung – Hamming, CRC, Reed-Solomon

$$\begin{array}{llll} C(x) = & x^3 + x^2 + 1 & = 1101 & \text{Generator} \\ S'(x) = & x^{10} + x^7 + x^6 + x^4 + x^2 + 1 & = 10011010101 & \text{empfangene Nachricht} \end{array}$$

Generator \swarrow

$$\begin{array}{r} 10011010101 \leftarrow S'(x) = \text{empfangene Nachricht inkl. CRC} \\ \underline{1101} \quad \text{XOR} \\ 1001 \\ \underline{1101} \\ 1000 \\ \underline{1101} \\ 1011 \\ \underline{1101} \\ 1100 \\ \underline{1101} \\ 1101 \\ \underline{1101} \\ 0 \end{array}$$

\swarrow Rest $S'(x) \bmod C(x)$

Ergebnis:

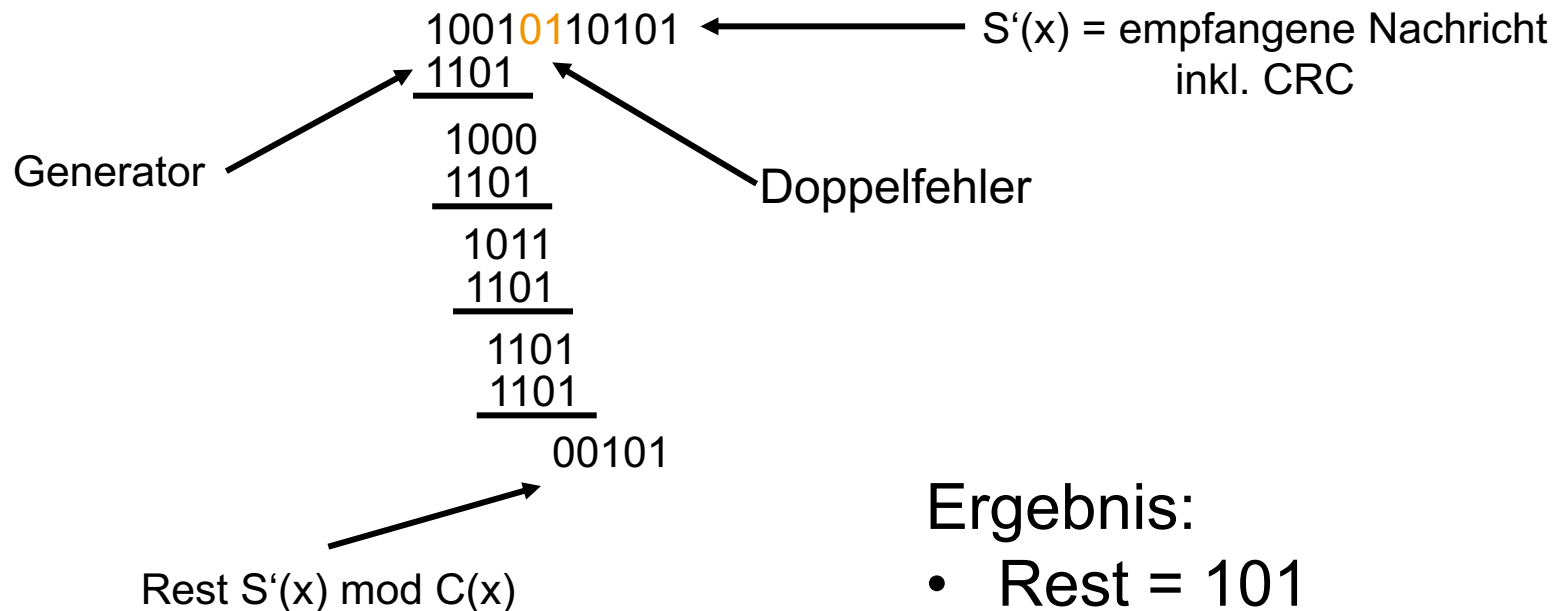
- Rest = 0
- CRC-Prüfung ok



CRC – Beispiel (Empfänger, mit Fehler)

Kapitel 4.2/4.3/4.4: Codesicherung und Kanalcodierung – Hamming, CRC, Reed-Solomon

$$\begin{array}{llll} C(x) = & x^3 + x^2 + 1 & = 1101 & \text{Generator} \\ S'(x) = & x^{10} + x^7 + x^5 + x^4 + x^2 + 1 & = 1001\textcolor{brown}{0}110101 & \text{empfangene Nachricht} \end{array}$$



Ergebnis:

- Rest = 101
- CRC-Prüfung nicht ok



- empfangen wird Polynom $S'(x) = S(x) + F(x)$
 - $F(x)$ ist ein Polynom, das die fehlerhaften Bit repräsentiert
 - $F(x) = 0 \rightarrow$ keine Fehler
- es können alle Fehler erkannt werden, bei denen $F(x)$ kein Vielfaches von $C(x)$ ist \rightarrow Anforderungen an Generator $C(x)$
- Welche Fehler können erkannt werden?
 - alle Einzelfehler, wenn x^k und der konstante Term 1 vorhanden sind
 - alle Doppelfehler, wenn $C(x)$ mindestens drei Terme hat, und die Größe der Daten kleiner als die Zykluslänge von $C(x)$ ist
 - alle r -Bit Fehler für ungerade r , wenn $C(x)$ eine gerade Anzahl an Termen hat; insbesondere, wenn es den Faktor $(x + 1)$ enthält
 - alle Burstfehler der Länge kleiner k , wenn $C(x)$ den konstanten Term enthält
 - die meisten Burstfehler der Länge $\geq k$



CRC – verbreitete Generatorpolynome

Kapitel 4.2/4.3/4.4: Codesicherung und Kanalcodierung – Hamming, CRC, Reed-Solomon

Name	Verwendung	Polynom
CRC-1	Paritätsbit	$x + 1$
CRC-4-CCITT	Telekommunikation = (15,11)-Hamming	$x^4 + x + 1$
CRC-5-USB	USB	$x^5 + x^2 + 1$
CRC-5-Bluetooth	Bluetooth	$x^5 + x^4 + x^2 + 1 =$ $(x^4 + x + 1)(x + 1)$
CRC-8-ITU-T	ISDN	$x^8 + x^2 + x + 1 =$ $(x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + 1)(x + 1)$
CRC-15-CAN	CAN-BUS	$x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1 =$ $(x^7 + x^3 + x^2 + x + 1)(x^7 + x^3 + 1)(x + 1)$
CRC-32	Ethernet, Serial ATA, ...	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} +$ $x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$



Zur Absicherung während der Übertragung sollen Daten mit einem CRC-Code versehen werden.

Die (binäre) zu sendende **Nachricht** lautet:

1100 0110

Als **Generatorpolynom** wird verwendet:

$$x^6 + x + 1$$

Wie lautet die zu sendende Nachricht inklusive des angehängten CRC-Codes?



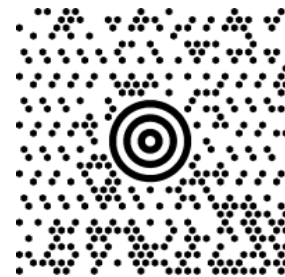
- viele verschiedene Varianten
- typisch:
 - unterschiedlich breite Punkte/Striche
 - dazwischen Lücken → hoher Kontrast zum Auslesen (z.B. mit Laserscanner oder Kamera)



Aztec-Code



DataMatrix-Code



MaxiCode

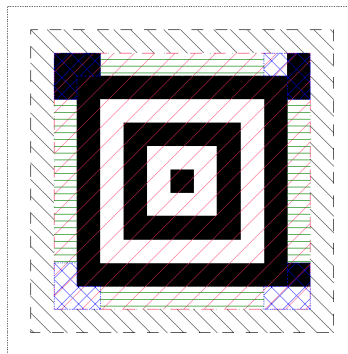


QR-Code

(Bilder aus Wikipedia)



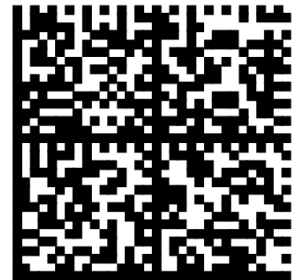
- entwickelt 1995, normiert in ISO/IEC 24778
- Verwendung: Online-Tickets
 - Deutsche/Schweizer/Österreichische Bahn
 - viele Fluggesellschaften
- kodiert 12 – 3000 Zeichen
- Reed-Solomon-Code zur Fehlerkorrektur
 - noch dekodierbar bei Zerstörung von bis zu 25%
- Zentrum: Markierung mit Orientierungspunkten



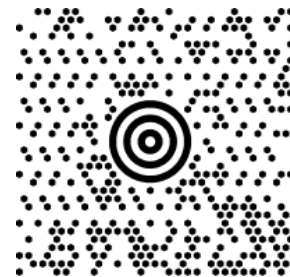
- entwickelt 1980er, normiert in ISO/IEC 16022
- Verwendung:
 - Beschriftung von Produkten mit Laser (dauerhaft)
 - Deutsche/Schweizer Post (Freimachung ohne Briefmarke)
- kodiert bis ca. 3000 Zeichen
- früher CRC-Code
- jetzt Reed-Solomon-Code
- rechteckiger Rand zum Finden des Codes und für Timing des Lesegeräts

STAMPIT
A00000CEE1

0,55 EUR
01.01.08



- 1989, normiert in ISO/IEC 16023
- Verwendung: UPS für Paketdaten
- kodiert 93 Zeichen
 - bis zu 8 Codes können kombiniert werden (→ 744 Zeichen)
- Reed-Solomon-Code zur Fehlerkorrektur
- Markierung in der Mitte
- hexagonale Punkte

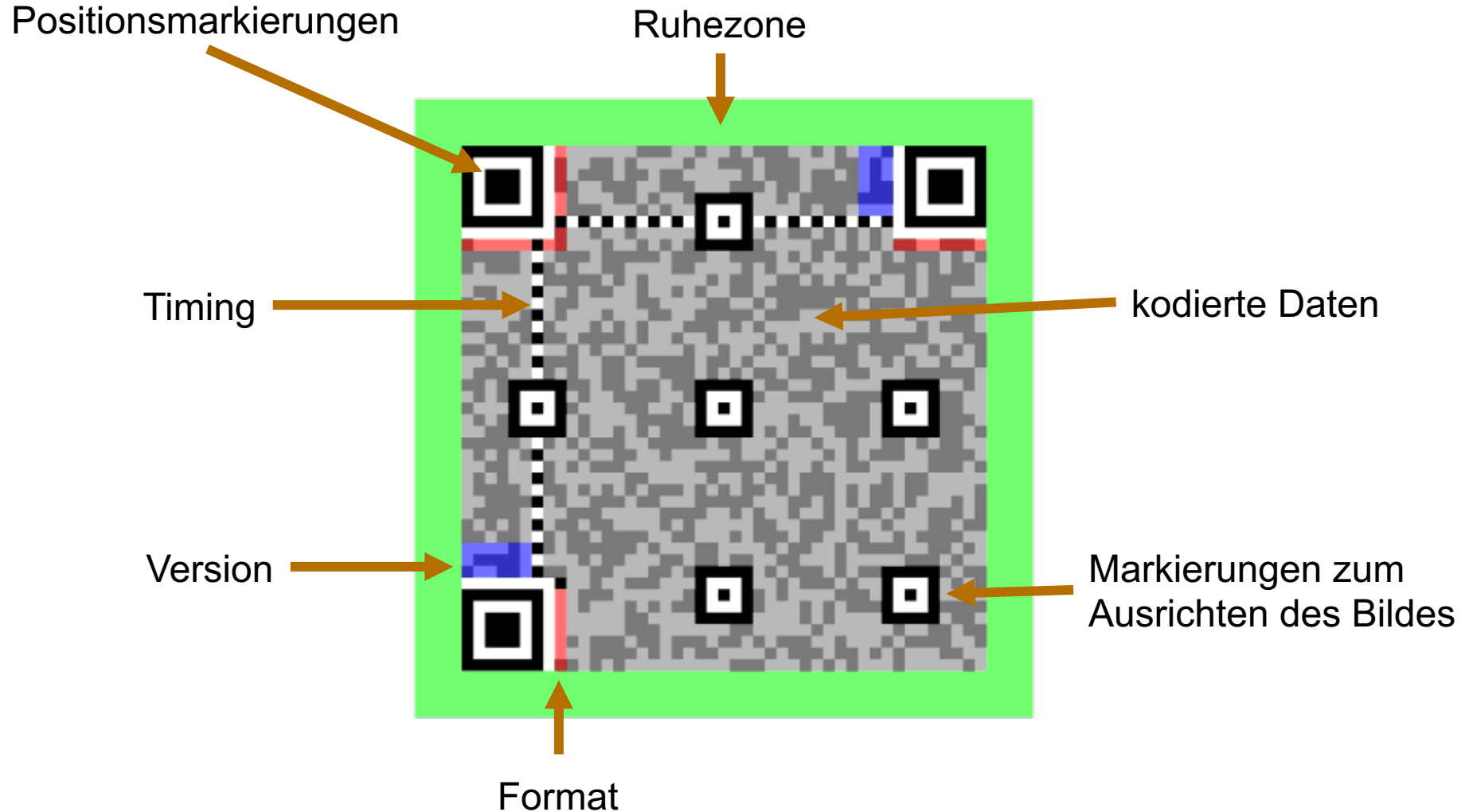


- QR = Quick Response
- 1994, entwickelt für Automotive-Bereich
- normiert in ISO/IEC 18004
- Verwendung:
 - ursprünglich industrielle Anwendungen
 - mittlerweile verbreitet bei Smartphones
- kodiert ca. 1800 – 7000 Zeichen
 - abhängig vom Modus
(nur Ziffern, lateinische Buchstaben, ganze Bytes,...)
 - und der gewünschten Robustheit gegen Fehler
 - bei mehr Daten: aufteilbar auf bis zu 16 Einzelcodes
- Reed-Solomon-Code zur Fehlerkorrektur
 - je nach Aufwand 7% – 30% der Daten rekonstruierbar
 - je robuster desto weniger Nutzdaten sind speicherbar



QR-Code – Aufbau

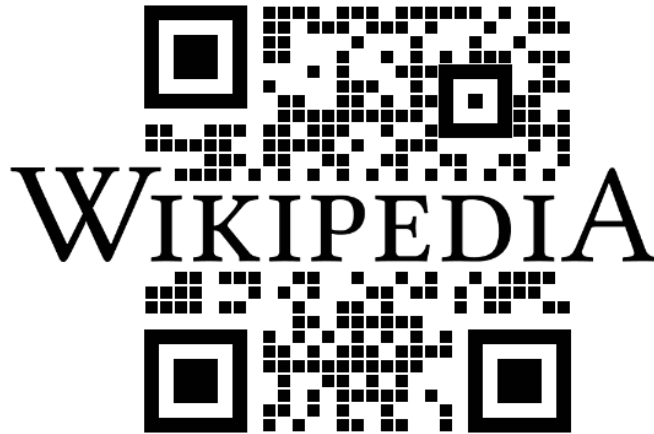
Kapitel 4.2/4.3/4.4: Codesicherung und Kanalcodierung – Hamming, CRC, Reed-Solomon



(Bild aus Wikipedia)



so etwas



geht nur wegen guter Fehlerkorrekturmechanismen
→ Reed-Solomon Codes



Reed-Solomon Codes (RS)

Kapitel 4.2/4.3/4.4: Codesicherung und Kanalcodierung – Hamming, CRC, Reed-Solomon

- Irving S. Reed und Gustave Solomon, 1960
- Eigenschaften:
 - Erkennung und Korrektur von
 - zufälligen Mehrfachfehlern
 - Burstfehlern
 - Auslöschungen (= fehlenden Daten)
 - nicht-binärer Code
 - also z.B. auf ASCII-Zeichen
 - wird zur eigentlichen Übertragung natürlich nach binär gewandelt
- Verwendung z.B.
 - QR-Codes, Audio-CD, DVD, Blu-Ray, Satelliten-Kommunikation, ...



- fasse Nachricht als *Koeffizienten eines Polynoms* über einem endlichen Körper auf
 - bei q Elementen = rechne modulo q (nur wenn q prim)
 - Koeffizienten können also nur Werte $0, 1, \dots, q - 1$ annehmen
- Codierung durch Auswertung des Polynoms an n verschiedenen Stellen
- Decodierung durch Interpolation



- Vorgehen zur Konstruktion von RS(q , m , n)
 - wähle endlichen Körper \mathbb{F}_q mit $q = p^l$ Elementen als Alphabet, p prim, $l \in \{1, 2, 3, \dots\}$
 - Nachricht (Block aus m Symbolen) $\mathbf{a} = (a_0, \dots, a_{m-1})$ aufgefasst als Polynom über \mathbb{F}_q :
$$P(x) = a_0 + a_1x + a_2x^2 + \dots + a_{m-1}x^{m-1}$$
 - wähle n paarweise verschiedene Elemente ($n \geq m$)
$$u_0, \dots, u_{n-1} \in \mathbb{F}_q$$
- Codierung
 - Auswertung von $P(x)$ an den n Stellen u_i
 - Hornerschema oder Diskrete Fourier-Transformation (DFT)
 - Codewort $\mathbf{c} = (P(u_0), P(u_1), \dots, P(u_{n-1}))$



- RS(q, m, n) mit $q = 5, m = 3, n = 5$
- Nachricht $\mathbf{a} = (1, 2, 1)$
 - Polynom: $P(x) = 1 + 2x + x^2$
- Auswertung von $P(x)$ an $n = 5$ Stellen
 - mehr geht nicht, da Körper nur 5 Elemente hat
 - $P(0) = 1 + 0 + 0 = 1$
 - $P(1) = 1 + 2 + 1 = 4$
 - $P(2) = 1 + 4 + 4 = 9 = 4 \mod 5$
 - $P(3) = 1 + 6 + 9 = 16 = 1 \mod 5$
 - $P(4) = 1 + 8 + 16 = 25 = 0 \mod 5$
- Codewort $\mathbf{c} = (1, 4, 4, 1, 0)$



- RS(q, m, n) toleriert bis zu $n - m$ Ausfälle
 - Ausfall:
 - ein Teil des Codes wurde nicht empfangen
 - Positionen der Ausfälle sind bekannt
 - es wurden also mindestens m Datenpunkte empfangen
- Polynom $P(x)$ hat Grad $m - 1$
 - aus m Datenpunkten lässt sich $P(x)$ rekonstruieren
 - und damit die Nachricht (= Koeffizienten von $P(X)$)
 - → Lagrange Interpolation



- gegeben: mindestens m Datenpunkte $(u_i, P(u_i))$
 - zur Vereinfachung der Notation:
Annahme, dass die ersten m empfangen wurden
- setze $g_i(x) = \prod_{j=0, j \neq i}^{m-1} (x - u_j)$, $i = 0, \dots, m - 1$
- es gilt $g_i(u_j) = 0, j \neq i$
- $P(x)$ erhält man aus

$$P(x) = \sum_{i=0}^{m-1} \frac{P(u_i)}{g_i(u_i)} g_i(x)$$



RS – Decodierung – Beispiel

Kapitel 4.2/4.3/4.4: Codesicherung und Kanalcodierung – Hamming, CRC, Reed-Solomon

- RS(q, m, n) mit $q = 5$, $m = 3$, $n = 5$ wie vorher
- Auswertung von $P(x)$ erfolgte an den Stellen 0, 1, 2, 3, 4
- gesendetes Codewort $\mathbf{c} = (1, 4, 4, 1, 0)$
 - letzte zwei Werte ausgefallen \rightarrow empfangen $(1, 4, 4, \varepsilon, \varepsilon)$
- Berechne Polynome $g_i(x)$:

$$\begin{aligned} g_0(x) &= (x - 1)(x - 2) = x^2 - 3x + 2 = x^2 + 2x + 2 \\ g_1(x) &= x(x - 2) = x^2 - 2x = x^2 + 3x \\ g_2(x) &= x(x - 1) = x^2 - x = x^2 + 4x \end{aligned} \quad \begin{array}{l} \swarrow \text{mod } 5! \end{array}$$



- Auswertung der $g_i(u_i)$ an den Stellen $u_i = 0, 1, 2$

$$g_0(x) = x^2 + 2x + 2$$

$$g_0(0) = 2$$

$$g_1(x) = x^2 + 3x$$

$$g_1(1) = 1 + 3 = 4$$

$$g_2(x) = x^2 + 4x$$

$$g_2(2) = 4 + 8 = 12 = 2$$

$$P(x) = \sum_{i=0}^{m-1} \frac{P(u_i)}{g_i(u_i)} g_i(x)$$



- Bestimmung der multiplikativen Inversen $g_i^{-1}(u_i)$
 - diese existieren immer, da Körper
 - verwende z.B. erweiterten euklidischen Algorithmus

$$g_0(0) = 2 \rightarrow g_0^{-1}(0) = 3 \quad (\text{Test: } 2 \cdot 3 = 6 = 1)$$

$$g_1(1) = 4 \rightarrow g_1^{-1}(1) = 4 \quad (\text{Test: } 4 \cdot 4 = 16 = 1)$$

$$g_2(2) = 2 \rightarrow g_2^{-1}(2) = 3 \quad (\text{Test: } 2 \cdot 3 = 6 = 1)$$

- Produkt $P(u_i)g_i^{-1}(u_i)$

$$P(0)g_0^{-1}(0) = 1 \cdot 3 = 3$$

$$P(1)g_1^{-1}(1) = 4 \cdot 4 = 16 = 1$$

$$P(2)g_2^{-1}(2) = 4 \cdot 3 = 12 = 2$$

$$P(x) = \sum_{i=0}^{m-1} \frac{P(u_i)}{g_i(u_i)} g_i(x)$$

(1, 4, 4, ε, ε)



- Ergebnis:

$$\begin{aligned} P(x) &= \sum_{i=0}^2 \frac{P(u_i)}{g_i(u_i)} g_i(x) = 3g_0(x) + 1g_1(x) + 2g_2(x) \\ &= 3(x^2 + 2x + 2) + (x^2 + 3x) + 2(x^2 + 4x) \\ &= 6x^2 + 17x + 6 \\ &= x^2 + 2x + 1 \end{aligned}$$

→ ursprüngliche Nachricht war (1, 2, 1)



- $RS(q, m, n)$ hat eine Hamming-Distanz von $n - m + 1$
- damit lassen sich also $(n - m) / 2$ Fehler korrigieren

Beweis:

- für $n \geq m$ können zwei Polynome nur an $m - 1$ Stellen die gleichen Werte haben
 - sonst wären sie identisch und die Nachrichten auch
 - die Werte der Polynome unterscheiden sich also an $n - m + 1$ Stellen
(= minimaler Abstand zwischen zwei Codewörtern)



- Man nehme zwei Polynome mit noch unbekannten Koeffizienten:
 - $f(x) = f_0 + f_1x + f_2x^2 + \dots$ vom Grad $\left\lceil \frac{n-m}{2} \right\rceil$
 - $g(x) = g_0 + g_1x + g_2x^2 + \dots$ vom Grad $\left\lceil \frac{n-m}{2} \right\rceil + m - 1$
- Konstruiere daraus ein neues Polynom
 - $p(x, y) = yf(x) + g(x)$
- Bestimme die Koeffizienten von $p(x, y)$ so, dass gilt
 - $p(u_i, y_i) = 0$,
wobei $y_i = P(u_i)$ die empfangene (fehlerhafte) Nachricht ist
- Die ursprünglich gesendete Nachricht ergibt sich aus den Koeffizienten des Polynoms
$$-\frac{g(x)}{f(x)}$$



RS – Decodierung – Beispiel

Kapitel 4.2/4.3/4.4: Codesicherung und Kanalcodierung – Hamming, CRC, Reed-Solomon

- RS(q, m, n) mit $q = 5, m = 3, n = 5$ wie vorher
 - $(n - m) / 2 = (5 - 3) / 2 = 1 \rightarrow 1$ Fehler korrigierbar
- Auswertung von $P(x)$ erfolgte an den Stellen 0, 1, 2, 3, 4
- gesendetes Codewort $\mathbf{c} = (1, 4, 4, 1, 0)$
 - eine Stelle falsch \rightarrow empfangen $(1, 4, 0, 1, 0)$
- Polynome:
 - $f(x) = f_0 + f_1x$ vom Grad $\left\lceil \frac{n-m}{2} \right\rceil = 1$
 - $g(x) = g_0 + g_1x + g_2x^2 + g_3x^3$ vom Grad $\left\lceil \frac{n-m}{2} \right\rceil + m - 1 = 3$
- ergibt
 - $p(x, y) = yf(x) + g(x) = f_0y + f_1xy + g_0 + g_1x + g_2x^2 + g_3x^3$



- $p(x, y) = yf(x) + g(x) =$
 $f_0y + f_1xy + g_0 + g_1x + g_2x^2 + g_3x^3$

- Paare (u_i, y_i) für $p(u_i, y_i) = 0$
empfangen $(1, 4, 0, 1, 0)$:
 $(0, 1), (1, 4), (2, 0), (3, 1), (4, 0)$

- Gleichungssystem:

$$\begin{aligned} f_0 + g_0 &= 0 \longrightarrow g_0 = -f_0 = 4f_0 && \swarrow \text{einsetzen in restliche} \\ 4f_0 + 4f_1 + g_0 + g_1 + g_2 + g_3 &= 0 && \text{Gleichungen +} \\ g_0 + 2g_1 + 4g_2 + 8g_3 &= 0 && \text{Reduktion mod 5} \\ f_0 + 3f_1 + g_0 + 3g_1 + 9g_2 + 27g_3 &= 0 \\ g_0 + 4g_1 + 16g_2 + 64g_3 &= 0 \end{aligned}$$

Achtung: Rechnung mod 5!



- ergibt

$$3f_0 + 4f_1 + g_1 + g_2 + g_3 = 0$$

$$4f_0 + 2g_1 + 4g_2 + 3g_3 = 0$$

$$3f_1 + 3g_1 + 4g_2 + 2g_3 = 0$$

$$4f_0 + 4g_1 + g_2 + 4g_3 = 0$$

- Lösen des Gleichungssystems

- z.B. mit Gauß-Elimination
- 5 Unbekannte, 4 Gleichungen → eine frei wählbar
- beachte: endlicher Körper, Inverse bzgl. Multiplikation:
 $1 \leftrightarrow 1, 2 \leftrightarrow 3, 3 \leftrightarrow 2, 4 \leftrightarrow 4$

- Ergebnis:

$$f_0 = 1, f_1 = 2, g_0 = 4, g_1 = 1, g_2 = 0, g_3 = 3$$



- Polynome:

- $f(x) = f_0 + f_1x = 1 + 2x$

- $g(x) = g_0 + g_1x + g_2x^2 + g_3x^3 = 4 + x + 3x^3$

- Berechne $\frac{g(x)}{f(x)}$

$$\begin{array}{r} (3x^3 + x + 4) : (2x + 1) = 4x^2 + 3x + 4 \\ - \underline{(3x^3 + 4x^2)} \\ (x^2 + x + 4) \\ - \underline{(x^2 + 3x)} \\ (3x + 4) \\ - \underline{(3x + 4)} \\ \text{-----} \end{array}$$

- Nachricht = $-\frac{g(x)}{f(x)} = -(4x^2 + 3x + 4) = x^2 + 2x + 1$
→ gesendet wurde (1, 2, 1)



- Decodierung in der Praxis
 - mit schnelleren (und komplizierteren) Verfahren
 - z.B. mit Berlekamp–Massey Algorithmus
- typische RS-Codes
 - CD: zwei hintereinander geschaltete RS-Codes
 - CIRC: Cross-Interleaved Reed-Solomon Coding
 - RS(33, 28, 32) und RS(29, 24, 28)
 - Burstfehler bis 4000 Bit (ca. 2,5mm Kratzer) exakt korrigierbar
 - Fehler hier = Ausfälle
 - DVD: ähnlich wie CD, aber größere Codes
 - RS(209, 192, 208) und RS(183, 172, 182)
 - QR: nicht lesbare Teile des Codes = Ausfälle

