

Inhaltsverzeichnis

1	Einführung	4
1.1	Inhalt	4
2	Kryptologie	5
2.1	Grundbegriffe und einfache Verfahren	5
2.1.1	Verschlüsselung erfordert	5
2.1.2	Beispiel für (nicht sicheres) symm. Verfahren	6
2.1.3	Prinzip von Kerkhoffs (1835-1903)	6
2.1.4	Arten von Angriffen	7
2.2	One-Time-Pad und perfekte Sicherheit	7
2.2.1	One-Time-Pad	7
2.2.2	Perfekte Sicherheit	7
2.3	Symmetrische Blockchiffre	8
2.3.1	Blockchiffre	8
2.3.2	Vorbemerkung	9
2.3.3	Affin-lineare Chiffren	10
2.4	Der Advanced Encryption Standard (AES)	12
2.4.1	Mathematische Methoden gebraucht für AES	12
2.4.2	SubBytes-Transfer	14
2.4.3	Shift Rows Transformation	15
2.4.4	Mix Columns Transformation	15
2.4.5	Schlüsselerzeugung	15
2.4.6	Public-Key-Systeme	16
2.4.7	Grundidee	16
2.5	RSA-Verfahren	16
2.5.1	Schlüsselerzeugung	17
2.5.2	Verschlüsselung	17
2.5.3	Entschlüsselung	17
2.5.4	Sicherheit vom RSA-Verfahren	19
2.5.5	Wie bestimmt man große Primzahlen?	19
2.5.6	Fermat-Test	20
2.5.7	Miller-Rabin-Test	20
2.5.8	Diffie-Hellman-Verfahren zur Schlüsselvereinbarung . . .	20

2.5.9	Sicherheit	21
2.5.10	Man-in-the-Middle	21
2.6	ElGamal-Public Key Verfahren (1984)	21
2.6.1	Schlüsselerzeugung	21
2.6.2	Verschlüsselung	21
2.6.3	Entschlüsselung	22
2.7	Signaturen, Hashfunktionen, Authentifizierung	22
2.7.1	Anforderung an digitale Signaturen	22
2.7.2	RSA-Signatur (vereinfachte Version)	22
2.7.3	Wie lassen sich lange RSA-Signaturen vermeiden?	22
2.7.4	RSA-Signatur mit HASH-Funktion	22
2.7.5	Angriffsmöglichkeiten	23
2.7.6	Satz: Geburtstagsparadoxon	23
2.7.7	Hashfunktion	24
2.7.8	Authentifizierung	24
2.7.9	Challenge-Response-Authentifizierung	25
2.8	Secret Sharing Scheme	25
2.8.1	(k, n) - Schwellenwertsysteme	25
3	Codierungstheorie	27
3.1	Grundbegriffe und einfache Beispiele	27
3.1.1	Codierung	27
3.1.2	Ziele	27
3.2	Grundprinzip	27
3.2.1	FEC-Verfahren (Forward Error Correction)	28
3.2.2	ARQ-Verfahren (Automatic Repeat Request)	28
3.2.3	Parity-Check-Codes	28
3.2.4	Wiederholungscode	28
3.2.5	(ehmaliger) ISBN-Code	29
3.2.6	EAN-13-Code	30
3.3	Blockcodes	32
3.3.1	Definition	32
3.3.2	Definition: Hamming-Abstand	32
3.3.3	Definition	33
3.4	Titel???	34
3.4.1	Definition: Perfekter Code	34
3.4.2	Gibt es perfekte Codes?	35
3.4.3	Lemma	35
3.4.4	Bsp: Binärer Hamming-Code der Länge 7	36
3.5	Lineare Codes	37
3.5.1	Bemerkung über endliche Körper	38
3.5.2	Bsp	38
3.5.3	Definition	38
3.5.4	Satz	38

3.5.5	Definition	39
3.5.6	Satz	39
3.5.7	Bemerkung	39
3.5.8	Bsp: Hamming-[7, 4]-Code über \mathbb{Z}_7	39
3.5.9	Def	39
3.5.10	Satz	39
3.5.11	Beweis	40
3.5.12	Bemerkung	40
3.5.13	Beispiel	40

Kapitel 1

Einführung

1.1 Inhalt

Übertragung (Speicherung) von Daten:

Schutz vor:

- zufälligen oder systematischen (physikalischen bedingten) Störungen
- Abhören, absichtliche Veränderung von Dritten (Kryptologie / Verschlüsselung)

Kryptologie:

- symmetrische Verfahren
- asymmetrische Verfahren (Public-Key Verfahren)
- Authentifizierung
- Signaturen

Codierungstheorie

- Fehlererkennung und Fehlerkorrektur
- lineare Blockcodes
- Decodierverfahren

Kapitel 2

Kryptologie

2.1 Grundbegriffe und einfache Verfahren

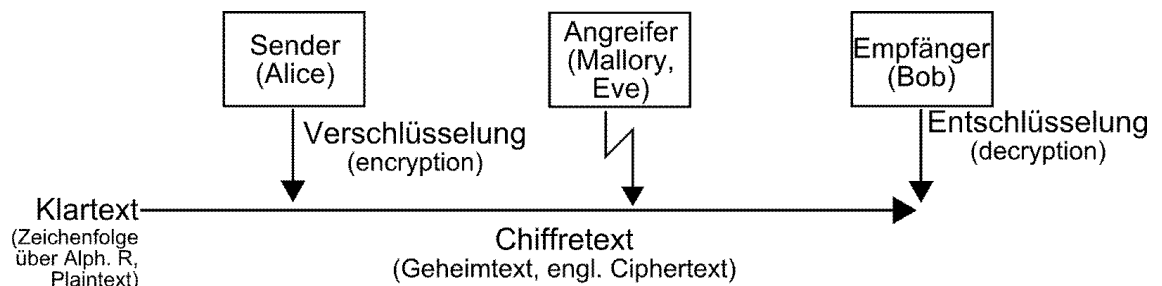


Abbildung 2.1: Schaubild der Kryptologie

2.1.1 Verschlüsselung erfordert

- Verschlüsselungsverfahren, Algorithmus (Funktion)
- Schlüssel k_e (encryption key)

$$E(m, k_e) = c$$

E =Verschlüsselungs Funktion, m =Klartext, c =Chiffretext

$$E(m_1, k_e) \neq E(m, k_e) \text{ für } m_1 \neq m_2$$

$$D(c, k_d) = m$$

(k_d zu k_e gehöriger Dechiffrierschlüssel!)

$k_d = k_e$ (oder k_d leicht aus k_e zu berechnen):

symmetrisches Verschl.verf., ansonsten **asymm. Verschl.verf.**. Ist k_d nur sehr schwer (oder garnicht) zu k_e berechenbar, so kann k_e veröffentl. werden:

Public-Key-Verfahren.

2.1.2 Beispiel für (nicht sicheres) symm. Verfahren

a) $R = S = \{0, 1, \dots, 25\}$

Verfahren: Verschiebechiffre

Schlüssel: $i \in \{0, 1, \dots, 25\}$

Verfahren $x \in \mathbb{R} \rightarrow x + i \bmod 26 = y$

$y \mapsto y - i \bmod 26 = x$

$m = x_1 \dots x_n \rightarrow c = (x_1 + i \bmod 26) \dots (x_n + i \bmod 26), E(m, i)$

Unsicher, weil Schlüsselmenge klein ist (Brute Force Angriff).

b) R, S , Schlüsselmenge=Menge aller Permutationen von $\{1, \dots, 25\} = S_{26}$

Verschl.: Wähle Permutation π

$x \in \mathbb{R} \rightarrow \pi(x) = y$

Entschl.: $y \rightarrow \pi^{-1}(y) = x$

$m = x_1 \dots x_r \rightarrow c = \pi(x_1) \dots \pi(x_r)$

$\begin{pmatrix} 0 & 1 & 2 & \dots & 25 \\ 3 & 17 & 4 & \dots & 13 \end{pmatrix} \rightarrow \pi(0) = 3, \text{ u.s.w.}$

Anzahl der Permutationen: $|S_{26}| = 26! \approx 4 \cdot 10^{26} \rightarrow$ Brute-Force Angriff nicht mehr möglich!

Warum? Man muss im Schnitt 50% der Permutationen testen. Angenommen man könnte 10^6 Perm. pro Sekunde testen.

Aufwand: $2 \cdot 10^{14}$ Sekunden $\approx 6.000.000$ Jahre

Trotzdem unsicher!

Grund: Charakteristische Häufigkeitsverteilung von Buchstaben in natürlichspr. Texten.

Verfahren beinhalten viele Verschlüsselungsmöglichkeiten, abhängig von der Auswahl des Schlüssels.

Verfahren bekannt, aber Schlüssel k_d geheim!

2.1.3 Prinzip von Kerkhoffs (1835-1903)

Sicherheit eines Verschlüsselungsverfahrens darf nicht von der Geheimhaltung des Verfahrens, sondern nur von der Geheimhaltung des verwendeten Schlüssels abhängen!

Kryptologie besteht aus Kryptographie (Entwurf) und der Kryptoanalyse (Angriff).
Angriffserfolge:

- Schlüssel k_d wird gefunden
- Eine zu der Dechiffrierfunktion $D(\cdot, k_d)$ äquivalente Funktion finden ohne Kenntnis von k_d
- gewisse Chiffretexte werden entschlüsselt

2.1.4 Arten von Angriffen

- Ciphertext-Only Angriff
- Known-Plaintext Angriff
- Chosen-Plaintext Angriff
- Chosen-Ciphertext Angriff

2.2 One-Time-Pad und perfekte Sicherheit

Lauftextverschlüsselung

Alphabet $\mathbb{Z}_k = \{0, 1, \dots, k-1\}$

In \mathbb{Z}_k kann man addieren und multiplizieren mit mod k .

Klartext x_1, x_2, \dots, x_n

Schlüsselwort k_1, k_2, \dots, k_n

$x_1 + k_1 \mod k, x_n + k_n \mod k \leftarrow$ Chiffretext

Mit natürlichsprachlichen Texten ist das Verfahren unsicher.

$\mathbb{Z}_2 = \{0, 1\}, 1 \oplus 1 = 0 = 0 \oplus 0, 0 \oplus 1 = 1 = 1 \oplus 0 \Rightarrow XOR$

Klartext in $\mathbb{Z}_2^n = \{(x_1, \dots, x_n) : x_i \in \mathbb{Z}_2\}$ Schlüssel: Zufallsfolge über \mathbb{Z}_2 der Länge n . m Klartext, k Zufallsfolge (beide Länge n)

$$c = m \oplus k, (x_1, \dots, x_n) \oplus (k_1, \dots, k_n) := (x_1 \oplus k_1, \dots, x_n \oplus k_n)$$

2.2.1 One-Time-Pad

Schlüssel k darf nur einmal verwendet werden!

$$m_1 \oplus k = c_1, m_2 \oplus k = c_2, c_1 \oplus c_2 = m_1 \oplus k \oplus m_2 \oplus k = m_1 \oplus m_2$$

Wieder nur Lauftext \rightarrow unsicher!

m_1 und m_2 lässt sich ermitteln.

Zufallsfolge der Länge n : eigentlich unsinniger Begriff. Da jedes Bit unabhängig von anderen mit Wahrscheinlichkeit $\frac{1}{2}$ erzeugt wird (Output einer binär symmetrischen Quelle)

Jede Folge der Länge n ist gleich wahrscheinlich (Wahrscheinlichkeit $\frac{1}{2^n}$)

One-Time-Pad ist perfekt sicher.

2.2.2 Perfekte Sicherheit

Ein Verschlüsselungsverfahren ist perfekt sicher, falls gilt: Für jeden Klartext m und jedem Chiffretext c (der festen Länge n)

$$pr(m|c) = pr(m)$$

$pr(m|c) \rightarrow$ A-posteriori-Wahrscheinlichkeit (Wahrscheinlichkeit, dass m Klartext, wenn c empfangen wurde)

$pr(m) \rightarrow$ A-priori-Wahrscheinlichkeit

Beispiel: Substitutionschiffre aus Kapitel 2.

$n = 5, m = HALLO, pr(m) > 0$

Ang: $c = QITUA$ wird empfangen, $LL \neq TU \rightarrow pr(m|c) = 0$

nicht perfekt sicher.

One-Time-Pad ist perfekt sicher.

(Bayes'sche Formel) $m \oplus k$

Jede Folge c lässt sich mit geeignetem k in der Form $c = m \oplus k$ erhalten.

Wähle $k = m \oplus c, m \oplus k = m \oplus m \oplus c = c$

Bei gegebenem m und zufällige gewählten Schlüssel k ist jeder Chiffretext gleichwertig.

2.3 Symmetrische Blockchiffre

2.3.1 Blockchiffre

Zerlege Klartext in Blöcke (Strings) der Länge n . Jeder Block wird einzeln verschlüsselt (in der Regel wieder in einem Block der Länge n). Gleiche Blöcke werden gleich verschlüsselt.

Wieviele Blockchiffren der Länge n gibt es?

Alphabet $\mathbb{Z}_2 = \{0, 1\}$

$|\{(0, \dots, 0), (0, \dots, 1), \dots, (1, \dots, 1)\}| = 2^n$

Block

Blockchiffre = Permutation der 2^n Blöcke.

$(2^n)!$ Blockchiffre

Wenn alle verwendet werden:

Schlüssel = Permutation der 2^n Blöcke

$(x_{1,1}, \dots, x_{1,n}, x_{2,1}, \dots, x_{2,n}, \dots)$ $n \cdot 2^n$ Bit

Zur Speicherung eines Schlüssels werden $n \cdot 2^n$ Bit benötigt.

Zum Beispiel:

$n = 64, 64 \cdot 2^{64} = 2^{70} \approx 1 \text{ ZetaByte} \approx 1 \text{ Milliarde Festplatten à 1 TB}$

Illusional!

Konsequenz: Verwende Verfahren, wo nur ein kleiner Teil der Permutation als Schlüssel verwendet wird und so sich die Schlüssel dann in kürzerer Form darstellt.

2.3.2 Vorbemerkung

$n \times m$ -Matrix

$$\begin{pmatrix} a_{11} & \dots & a_{1m} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nm} \end{pmatrix}$$

$1 \times n$ = Zeilenvektor $= (a_1, \dots, a_n)$

$n \times 1$ = Spaltenvektor $= \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$

z.B. $a_{ij} \in \mathbb{R}$, $a_{ij} \in \mathbb{Z}$ oder $a_{ij} \in R$, R Ring

$n \times m$ -Matrix A, B

$$\begin{pmatrix} a_{11} & \dots & a_{1m} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nm} \end{pmatrix} + \begin{pmatrix} b_{11} & \dots & b_{1m} \\ \vdots & & \vdots \\ b_{n1} & \dots & b_{nm} \end{pmatrix} := \begin{pmatrix} a_{11} + b_{11} & \dots & a_{1m} + b_{1m} \\ \vdots & & \vdots \\ a_{n1} + b_{n1} & \dots & a_{nm} + b_{nm} \end{pmatrix}$$

$$A = n \times m, B = m \times k,$$

$$A \cdot B = \begin{pmatrix} c_{11} & \dots & c_{1k} \\ \vdots & & \vdots \\ c_{m1} & \dots & c_{mk} \end{pmatrix} = n \times k$$

$$c_{1l} = (a_{11} \cdot b_{1l}) + (a_{12} \cdot b_{2l}) + \dots + (a_{1m} \cdot b_{ml})$$

$$(A + B) \cdot C = A \cdot C + B \cdot C$$

Im Allgemeinen: $A \cdot B \neq B \cdot A$

Quadratische Matrix ($n \times n$)

$$E_n = \begin{pmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{pmatrix}$$

$$A = n \times n, A \cdot E_n = E_n \cdot A = A$$

A $n \times n$ -Matrix über kommutativen Ring R mit Eins.

Wann existiert Matrix A^{-1} (Inverse Matrix) mit $A^{-1} \cdot A = A \cdot A^{-1} = E_n$?

$\det(A) \in R$ Determinante von A

$$2 \times 2\text{-Matrix: } \det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = a_{11} \cdot a_{22} - a_{12} \cdot a_{21}$$

A besitzt inverse Matrix $\Leftrightarrow \det(A) \neq 0$ in R ein inverses besitzt

(z.B. \mathbb{R} Körper, $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}_p$, $\det(A) \neq 0$)

$$A^{-1} = \begin{pmatrix} \frac{1}{\det(A)} \cdot b_{11} & \dots & \frac{1}{\det(A)} \cdot b_{1m} \\ \vdots & & \vdots \\ \frac{1}{\det(A)} \cdot b_{n1} & \dots & \frac{1}{\det(A)} \cdot b_{nm} \end{pmatrix}$$

$$b_{ij} = (-1)^{i+j} \det(A_{ji})$$

$A_{ji} = (n-1) \times (n-1)$ -Matrix, die aus A durchstreichen der j -ten Zeile und i -ten Spalte entsteht.

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad A^{-1} = \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}$$

$$R = \mathbb{Z}_k \{0, 1, \dots, k\}$$

Addition und Multiplikation in $\mathbb{Z}_k(\oplus, \odot)$

normale Add. und Mult. mit mod k

2.3.3 Affin-lineare Chiffren

Klartextalphabet = Chiffretextalphabet = \mathbb{Z}_k ($k = 2, k = 26$)

Wähle $n \times n$ -Matrix A über \mathbb{Z}_k und Zeilenvektor b der Länge n über \mathbb{Z}_k . Dies wird der Schlüssel sein für die Chiffrierung.

Blockchiffre der Länge n .

Block = Zeilenvektor der Länge n über \mathbb{Z}_k .

Klartextblock v

Chiffretextblock

$$v \cdot A + b =: w$$

$$v \rightarrow v \cdot A + b =: w$$

$$w - b = v \cdot A$$

benötigen: A^{-1} existiert (d.h. $\text{ggT}(\det(A), k) = 1$)

Dechiffrierung:

$$(w - b) \cdot A^{-1} = v \cdot A \cdot A^{-1} = v \cdot E_n = v$$

(wenn immer $b=0$ gewählt wird, dann lineare Chiffren, Hill-Chiffren)

Beispiel:

$$A = \begin{pmatrix} 1 & 3 \\ 3 & 2 \end{pmatrix}, \quad \mathbb{Z}_6$$

Blockchiffre der Länge n $\det(A) = 1 \cdot 2 - 3 \cdot 3 = -7 = 5$ inverse in \mathbb{Z}_6

$$\frac{1}{\det(A)} = \det(A)^{-1} = 5$$

$$A^{-1} = 5 \cdot \begin{pmatrix} 2 & -3 \\ -3 & 1 \end{pmatrix} = \begin{pmatrix} 10 & -15 \\ -15 & 5 \end{pmatrix} = \begin{pmatrix} 4 & 3 \\ 3 & 5 \end{pmatrix}$$

Test:

$$A \cdot A^{-1} = \begin{pmatrix} 1 & 3 \\ 3 & 2 \end{pmatrix} \cdot \begin{pmatrix} 4 & 3 \\ 3 & 5 \end{pmatrix} = \begin{pmatrix} 4+9 & 3+15 \\ 12+6 & 9+10 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Verschlüsselung: Schlüssel:

$$A = \begin{pmatrix} 1 & 3 \\ 3 & 2 \end{pmatrix} \quad b = (3, 5)$$

Klartextblock: $(1, 2)$

Chiffretextblock:

$$w = (1, 2) \cdot \begin{pmatrix} 1 & 3 \\ 3 & 2 \end{pmatrix} + (3, 5) = (1, 1) + (3, 5) = (4, 0)$$

Entschlüsselung:

$$(w - b) \cdot A^{-1} = (1, 1) \cdot \begin{pmatrix} 4 & 3 \\ 3 & 5 \end{pmatrix} = (1, 2)$$

$\mathbb{Z}_2 : n^2 + n$ Bit zur Speicherung eines Schlüssels.

Wieviele inverse Matrizen über \mathbb{Z}_2 mit $n = 64$?

$$(2^{64} - 1) \cdot (2^{64} - 2) \cdot \dots \cdot (2^{64} - 2^{63}) \approx 0.29 \cdot 2^{4096}$$

Verfahren ist unsicher gegenüber Known-Plaintext-Angriffe.

(A, b) Schlüssel, A inverse $n \times n$ -Matrix über \mathbb{Z}_k , $b \in \mathbb{Z}_k^n$

Angenommen Angreifer kennt $n + 1$ Klartext/Chiffretextpaare verschlüsselt mit

(A, b) , v_0, v_1, \dots, v_n w_0, \dots, w_n

Dann kann er häufig (A, b) bestimmen.

$$V = \begin{pmatrix} v_1 - v_0 \\ v_2 - v_0 \\ \vdots \\ v_n - v_0 \end{pmatrix} \quad n \times n - \text{Matrix}$$

Angenommen: V ist invertierbar. Setze $W = \begin{pmatrix} w_1 - w_0 \\ \vdots \\ w_n - w_0 \end{pmatrix}$

$$V \cdot A = \begin{pmatrix} (v_1 - v_0) \cdot A \\ \vdots \\ (v_n - v_0) \cdot A \end{pmatrix} = \begin{pmatrix} v_1 \cdot A + b - v_0 \cdot A + b \\ \vdots \\ v_n \cdot A + b - v_0 \cdot A + b \end{pmatrix} = \begin{pmatrix} w_1 - w_0 \\ \vdots \\ w_n - w_0 \end{pmatrix} = W$$

$V \cdot A$ bekannt, also auch V^{-1} :

$$A = V^{-1} \cdot w$$

$$b = w_0 - v_0 \cdot A$$

Beispiel: $n = 2$, $k = 25$ $\{A, \dots, Z\} = \{0, \dots, 25\}$

HERBST			→ NEBLIG		
H	7	v_0	N	13	w_0
E	4		E	4	
R	17	v_1	B	1	w_1
B	1		L	11	
S	18	v_2	I	8	w_2
T	19		G	6	

$$V = \begin{pmatrix} 10 & -3 \\ 11 & 15 \end{pmatrix} = \begin{pmatrix} 10 & 23 \\ 11 & 15 \end{pmatrix}, \quad W = \begin{pmatrix} 14 & 7 \\ 21 & 2 \end{pmatrix}$$

$$\det(V) = 10 \cdot 15 + 33 = 183 \equiv 1 \pmod{26}$$

$$V^{-1} = \begin{pmatrix} 15 & 3 \\ -11 & 10 \end{pmatrix} = \begin{pmatrix} 15 & 3 \\ 15 & 10 \end{pmatrix}$$

$$A = V^{-1} \cdot W = \begin{pmatrix} 15 & 3 \\ 15 & 10 \end{pmatrix} \cdot \begin{pmatrix} 14 & 7 \\ 21 & 2 \end{pmatrix} = \begin{pmatrix} 210 + 63 & 105 + 6 \\ 210 + 210 & 105 + 20 \end{pmatrix} = \begin{pmatrix} 13 & 7 \\ 4 & 21 \end{pmatrix}$$

$$b = w_0 - v_0 \cdot A = (13, 4) - (7, 4) \cdot \begin{pmatrix} 13 & 7 \\ 4 & 21 \end{pmatrix} = (10, 1)$$

Test:

$$v_1 \cdot A + b = w_1, \quad v_2 \cdot A + b = w_2$$

2.4 Der Advanced Encryption Standard (AES)

2.4.1 Mathematische Methoden gebraucht für AES

Seit 70er Jahren gab es DES (Blocklänge 64 Bit, Schlüssellänge 56 Bit)

Nachfolger des DES: Daemen, Rijmen (Belgier)

Rijndael-Verfahren → AES (2002 FIPS 197)

Iterierte Blockchiffre

Version mit 128 Bit Block und Schlüssellänge.

Vorbemerkung: 128-Bit Blöcke werden dargestellt als:

$$\begin{pmatrix} a_{01} & a_{02} & \dots & a_{03} \\ a_{10} & a_{11} & \dots & a_{13} \\ \vdots & \vdots & \ddots & \vdots \\ a_{30} & \dots & \dots & a_{33} \end{pmatrix}$$

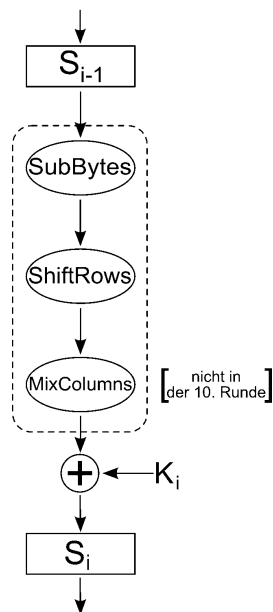


Abbildung 2.2: Eine Runde vom AES

Jedes a_{ij} = Byte

128er Block $\hat{=} a_{00}a_{10}a_{20} \dots a_{01}a_{11} \dots a_{33}$ (spaltenweise gelesen)

endlicher Körper: einfachste Möglichkeit \mathbb{Z}_p (p Primzahl)

\mathbb{F}_{2^8} Körper mit $2^8 = 256$ Elementen

Menge: Polynome vom Grad < 8 über \mathbb{Z}_2

$b_7x^7 + \dots + b_1x + b_0, b_i \in \mathbb{Z}_2$

(b_7, b_6, \dots, b_0) Byte

Addition = normale Addition von Polynomen

Multiplikation = normale Multiplikation von Polynomen + Reduktion modulo irreduzibler Polynom vom Grad 8. $(x^8 + x^4 + x^3 + x + 1)$

Bsp.

$$(x^7 + x + 1) \odot (x^3 + x) = x^{10} + x^8 + x^4 + x^3 + x^2 + x$$

$$x^{10} + x^8 + x^4 + x^3 + x^2 + x \mod x^8 + x^4 + x^3 + x + 1$$

$$x^{10} + x^8 + x^4 + x^3 + x^2 + x \div x^8 + x^4 + x^3 + x + 1 = x^2 + 1$$

$$\begin{array}{r} x^{10} + x^8 + x^4 + x^3 + x^2 + x \\ \underline{x^{10} + x^6 + x^5 + x^3 + x^2} \\ x^8 + x^6 + x^5 + x^4 + x \\ \underline{x^8 + x^4 + x^3 + x + 1} \end{array}$$

$$x^6 + x^5 + x^3 + 1 \leftarrow$$

$$(x^7 + x + 1) \odot (x^3 + x) = x^6 + x^5 + x^3 + 1$$

In \mathbb{F}_{2^8} hat jedes Element $\neq 0$ ein Inverses bzgl. \odot :

$$g \neq 0. \exists x. g \cdot x^{-1} \in \mathbb{F}_{2^8} : g \odot g^{-1} = 1$$

Erweiterte Euklid. Algo. für Polynome:

$$g \neq 0 \text{ (Grad} \leq 7) \quad h = x^8 + x^4 + x^3 + x + 1 \text{ irred.} \quad \text{ggT}(g, h) = 1$$

$$\text{EEA: } u, v \in \mathbb{Z}_2[x] : u \cdot g + v \cdot h = 1$$

$$u \bmod h =: g^{-1}$$

$$g^{-1} \odot g = ((u \bmod h) \cdot g) \bmod h = u \cdot g \bmod h = (1 - vh) \bmod h = 1 \bmod h = 1$$

2.4.2 SubBytes-Transfer

$$S_{i-1} = \begin{pmatrix} a_{01} & a_{02} & \dots & a_{03} \\ a_{10} & a_{11} & \dots & a_{13} \\ \vdots & \vdots & \ddots & \vdots \\ a_{30} & \dots & \dots & a_{33} \end{pmatrix}, a_{ij} \text{ Bytes}$$

Sei g eines dieser Bytes, $g = (b_7 b_6 \dots b_0)$, $b_i \in \mathbb{Z}_2$

1. Schritt: Fasse g als Element in \mathbb{F}_{2^8} auf.

Ist $g = (0, \dots, 0)$, so lasse g unverändert.

Ist $g \neq (0, \dots, 0)$, so ersetze g durch g^{-1} .

2. Schritt: Ergebnis nach Schritt 1: \tilde{g} wird folgenderm. transformiert

$$\tilde{g} \cdot A + b = \tilde{\tilde{g}} \quad (\text{affin-lin. Transformation}) \quad (\tilde{g}: g\text{-schlange}, \tilde{\tilde{g}}: g\text{-doppel-schlange})$$

A wird durch zyklischer Shift der vorherigen Zeile um 1 Stelle nach rechts erzeugt.

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \quad b = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Schritt 1 und 2 werden kombiniert, nicht jedes mal berechnet. Alle möglichen Sub-Bytes (2^8 viele) sind in einer 16x16 Matrix und wird per Table-Lookup nachgeschlagen.

$$g = (b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0) \quad b_7 b_6 b_5 b_4 = 0 \text{ bis } 15 \text{ (Zeile)} \quad b_3 b_2 b_1 b_0 \text{ (Spalte)}$$

2.4.3 Shift Rows Transformation

4x4-Matrix von Bytes:

$$\begin{pmatrix} & & & \\ & & & \\ & & & \\ & & & \end{pmatrix} \begin{array}{l} \leftarrow \text{Erste Zeile unverändert} \\ \leftarrow 1 \text{ Stelle nach links cykl.} \\ \leftarrow 2 \text{ Stellen nach links cykl.} \\ \leftarrow 3 \text{ Stellen nach links cykl.} \end{array}$$

2.4.4 Mix Columns Transformation

4x4-Matrix, Einträge als Elemente in \mathbb{F}_{2^8} auffassen.

Multiplikation von links mit Matrix (Mult. der Eintr. in \mathbb{F}_{2^8}):

$$\begin{pmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{pmatrix}$$

$$x \triangleq (0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0)$$

2.4.5 Schlüsselerzeugung

Ausgangsschlüssel hat 128 Bit. (16er String in Hexcode)

Schreibe als 4x4-Matrix von Bytes. 4 Spalten $w(0), w(1), w(2), w(4)$. Definiere weitere 40 Spalten à 4 Bytes.

$w(i-1)$ sei schon definiert.

$$4 \nmid i : w(i) := w(i-4) \oplus w(i-1) \quad (\text{byteweise XOR})$$

$$4 \mid i : w(i) := w(i-4) \oplus T(w(i-1)) \quad (T \text{ Transformation})$$

T?

$$w(i-1) = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}, \quad a, \dots, d \text{ Bytes}$$

Wende auf b, c, d, a SubBytes-Transformation an $\rightarrow e, f, g, h$

$$r(i) = (0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0)^{\frac{(i-4)}{4}} \quad \text{Potenz. in } \mathbb{F}_{2^8}$$

$$T(w(i-1)) = \begin{pmatrix} e \oplus r(i) \\ f \\ g \\ h \end{pmatrix}$$

Rundenschlüssel K_i : 4x4-Matrix mit Spalten $w(4i), ; w(4i+1), ; w(4i+2), ; w(4i+3)$

(Nebenbemerkung: Linear heißt $f(x+y) = f(x) + f(y)$)

2.4.6 Public-Key-Systeme

2.4.7 Grundidee

Diffie, Hellman, 1976 Jeder Teilnehmer hat ein Paar von Schlüsseln:

- Öffentlichen Schlüssel P_A
- geheimen Schlüssel G_A

Zu P_A gehört öffentlich bekannte Verschlüsselungsfunktion $E_{P_A} (=E(\cdot, P_A))$

$$B \xrightarrow{m} A : E_{P_A}(m) = c$$

1. m darf mit „realistischen Aufwand“ nicht aus $E_{P_A}(m)$ berechenbar sein. E_{P_A} ist **Einwegfunktion**
(E_{P_A} muss effizient berechenbar sein, aber $E_{P_A}^{-1}$ nicht!)
2. A muss mit Hilfe einer Zusatzinformation ($=G_A$) in der Lage sein, $E_{P_A}^{-1}$ effizient zu berechnen.

$$D_{G_A}(c) = m = E_{P_A}^{-1}(c)$$

Injektive Einwegfunktionen, die mit Zusatzinformation effizient invertierbar sind: **Geheimtürfunktion** (trapdoor function)

Aus 1) und 2) folgt:

3. G_A darf aus P_A nicht schnell berechenbar sein!

Es ist unbekannt ob Einwegfunktion existieren! Notwendig für die Existenz von Einwegfunktionen:

$$P \neq NP$$

Es gibt Kandidaten für Einwegfunktionen.

2.5 RSA-Verfahren

(Rivest, Shamir, Adleman, 1977) Beruht auf Schwierigkeit große Zahlen zu faktorisieren!

2.5.1 Schlüsselerzeugung

Wähle zwei große Primzahlen p, q ($p \neq q$) (mindestens 500 Bit Länge)

Bilde $n = p \cdot q$

$$\varphi(n) = ||\{a \in \mathbb{N} : 1 \leq a < n, \text{ggT}(a, n) = 1\}||$$

$$n = p \cdot q : \varphi(n) = (p - 1) \cdot (q - 1)$$

[nicht teilerfremd zu n : $1 \cdot p, 2 \cdot p, \dots, (q - 1) \cdot p = n = 1 \cdot q, 2 \cdot q, \dots, (p - 1) \cdot q$

$(p - 1) + (q - 1) + 1$

$$\varphi(n) = n - (p - 1) - (q - 1) - 1 = n - p - q + 1 = p \cdot q - p - q + 1 = (p - 1) \cdot (q - 1)]$$

Wähle e , $1 < e < \varphi(n)$ mit $\text{ggT}(e, \varphi(n)) = 1$

Zufallswahl, bestimme $\text{ggT}(e, \varphi(n))$ mit Euklidischer Algorithmus, so lange, bis e mit $\text{ggT}(e, \varphi(n)) = 1$ gefunden ist.)

öffentlicher Schlüssel

$$(n, e) = P_A$$

Wähle $d < \varphi(n)$ mit $e \cdot d \equiv 1 \pmod{n}$ (d.h. $\varphi(n) \mid e \cdot d - 1, e \cdot d = 1 + k \cdot \varphi(n)$ für $k \in \mathbb{N}$)

(Wende erweiterten Euklidischen Algorithmus auf $e, \varphi(n)$ an:

Liefert $u, v \in \mathbb{Z}$ mit $u \cdot e + v \cdot \varphi(n) = \text{ggT}(e, \varphi(n)) = 1$

$$d = u \pmod{\varphi(n)}$$

$$u \cdot e + v \cdot \varphi(n) \pmod{\varphi(n)} = 1$$

$$\underbrace{(u \pmod{\varphi(n)} \cdot e)}_d \pmod{\varphi(n)} = 1$$

Geheimerschlüssel

$$G_A = d$$

2.5.2 Verschlüsselung

B Nachricht an A . Codiere Nachricht als Zahl. Zerlege in Blöcke deren Zahlwert $< n$. Sei m so ein Block. ($m < n$)

$$m^e \pmod{n} = c$$

2.5.3 Entschlüsselung

$$c^d \pmod{n} = m$$

Gültigkeit basiert auf kleinem Satz von Fermat:

r Primzahl, $\text{ggT}(a, r) = 1$ (d.h. $r \nmid a$)

$$a^{r-1} \equiv 1 \pmod{r}$$

Sei $m < n = p \cdot q$

$$\begin{aligned} c &= m^e \pmod n, \quad c^d \pmod n = m^{e \cdot d} \pmod n \\ e \cdot d &= 1 + k \cdot \varphi(n) \\ &= 1 + k \cdot (p-1) \cdot (q-1) \end{aligned}$$

Ist $p \nmid m$, so

$$\begin{aligned} m^{e \cdot d} &= m^{1+k \cdot (p-1) \cdot (q-1)} \\ &= m \cdot \underbrace{(m^{p-1})^{k \cdot (q-1)}}_{\equiv 1 \pmod p} \\ &\stackrel{\text{mod } p}{\Rightarrow} m \cdot 1^{k \cdot (q-1)} \pmod p \\ &\equiv m \pmod p \end{aligned}$$

Ist $p \mid m$:

$$m \equiv 0 \equiv m^{e \cdot d} \pmod p$$

In jedem Fall:

$$m^{e \cdot d} \equiv m \pmod p$$

Genauso:

$$\begin{aligned} m^{e \cdot d} &\equiv m \pmod q \\ p \mid m^{e \cdot d} - m, \quad q \mid m^{e \cdot d} - m, \quad p \neq q &\Rightarrow n = p \cdot q \mid m^{e \cdot d} - m \\ m^{e \cdot d} &\equiv m \pmod n, \quad m^{e \cdot d} \pmod n = m \end{aligned}$$

Schnelle Berechnung von modularen Potenzen ($m^e \pmod n$)

$$\begin{aligned} e &= \sum_{i=0}^k e_i \cdot 2^i, \quad e_i \in \{0, 1\}, \quad e_k = 1 \\ m^e &= m^{2 \cdot k + e_{k-1} \cdot 2^{k-1} + \dots + e_1 \cdot 2 + e_0} \\ &= ((\dots ((m^2 \cdot m^{e_{k-1}})^2 \cdot m^{e_{k-2}})^2 \dots)^2 \cdot m^{e_1})^2 \cdot m^{e_0} \end{aligned}$$

gelöst im worst case mit $2 \cdot k$ Multiplikationen.

$$k = \lfloor \log_2(e) \rfloor$$

Nach jedem Rechenschritt mod n reduzieren!

2.5.4 Sicherheit vom RSA-Verfahren

Falls p, q bekannt $\Rightarrow \varphi(n), d$ bekannt.

$\varphi(n)$ bekannt $\Rightarrow p, q$ bekannt.

$\varphi(n) = n - q - p + 1$ bekannt $\Rightarrow p + q = s$ bekannt, $p \cdot q = n$ bekannt.

$p \cdot (s - p) = n \Rightarrow p^2 - s \cdot p + n = 0$ quadratische Gleichung für p

Es gilt auch: Bestimmung von d ist „genauso schwierig“ wie die Faktorisierung von n .

Komplexität der besten Faktorisierungsalgorithmen:

$$O(e^{c \cdot (\log n)^{\frac{1}{3}} \cdot ((\log \log n)^{\frac{2}{3}})})$$

Um eine 640 Bit Zahl zu faktorisieren braucht man 30-CPU-Jahre auf einer 2.2 GHz CPU.

Häufig wird $e = 3$ gewählt.

$$A \xrightarrow{m} (n_1, 3) B_1$$

$$A \xrightarrow{m} (n_2, 3) B_2$$

$$A \xrightarrow{m} (n_3, 3) B_2$$

$$\text{ggT}(n_i, n_j) = 1$$

$$c_1 = m^3 \mod n_1$$

$$c_2 = m^3 \mod n_2$$

$$c_3 = m^3 \mod n_3$$

Eve fängt c_1, c_2, c_3 ab: Chinesischer Restsatz:

$$0 \leq x \leq n_1 \cdot n_2 \cdot n_3$$

$$\text{mit } x = c_i \mod n_i, \quad i = 1, 2, 3$$

x ist eindeutig bestimmbar

$$m^3 \equiv c_i \mod n_i, \quad m^3 < n_1 \cdot n_2 \cdot n_3$$

$$\Rightarrow x = m^3 \Rightarrow m = \sqrt[3]{x}$$

Wenn $e = 5$, dann braucht man 5 Nachrichten.

2.5.5 Wie bestimmt man große Primzahlen?

$$p \text{ Primzahl, } a \in \mathbb{Z}, \quad \text{ggT}(a, p) = 1$$

$$a^{p-1} \equiv 1 \pmod{p} \text{ [kl. Satz von Fermat]}$$

gegeben: $n, \quad \text{ggT}(a, n) = 1 \quad a^{n-1} \equiv 1 \pmod{n}?$

2.5.6 Fermat-Test

Wenn nicht, so ist n keine Primzahl. Wenn ja, so keine Aussage möglich. Wähle neues a !

Es gibt zusammengesetzte Zahlen n (Carmichael-Zahlen) mit:

$$a^{n-1} \equiv 1 \pmod{n} \quad \forall a \text{ mit } \text{ggT}(a, n) = 1$$

2.5.7 Miller-Rabin-Test

$$\text{ggT}(a, p) = 1$$

$$p \text{ Primzahl } p-1 = 2^s \cdot t, \quad 2 \nmid t$$

$$a^{2^s \cdot t} \equiv 1 \pmod{p}$$

$$(a^{2^{s-1} \cdot t})^2 = b$$

$$a^{2^{s-1} \cdot t} = \begin{cases} 1 & \pmod{p} \\ -1 & \pmod{p} \end{cases}$$

$$b^2 \equiv 1 \pmod{p}$$

$$(b \pmod{p})^2 = 1 \in \mathbb{Z}_p$$

$$x^2 - 1 \in \mathbb{Z}_p[x]$$

Entweder $a^i \equiv 1 \pmod{p}$ oder $a^{s^i \cdot t} \equiv -1 \pmod{p}$ für ein $0 \leq i \leq s$ Teste dies mit n statt p .

Wenn n keine Primzahl ist, dann gibt es mindestens $\frac{3}{4}\varphi(n)$ viele a , so dass der Test fehlschlägt.

→ probabilistischer Primzahltest

p Primzahl $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$ Gruppe bezüglich Multiplikation (zyklisch)

$$\exists g \in \mathbb{Z}_p^* : \{g^0, g^1, g^2, \dots, g^{p-2}\} = \mathbb{Z}_p^*$$

$$g^{p-1} \equiv 1 \pmod{p}$$

Primitivwurzel \pmod{p}

$0 \leq a \leq p-2 : a \mapsto g^a \pmod{p}$ Kandidat für Einwegfunktion.

$g^a \pmod{p} \rightarrow a$ (diskreter Logarithmus) ist nach heutigem Stand schwer!

2.5.8 Diffie-Hellman-Verfahren zur Schlüsselvereinbarung

A, B wollen gemeinsamen Schlüssel K für ein symm. Verfahren vereinbaren; es steht nur unsichere Kommunikationskanal zur Verfügung.

Lösung: p, g (Bitlänge von $p >$ Bitlänge von K) (können öffentlich bekannt sein).

1. A wählt zufällig $a \in \{2, \dots, p-2\}$
 A berechnet $x = g^a \mod p$
 (a geheim halten)

2. B wählt zufällig $b \in \{2, \dots, p-2\}$
 B berechnet $y = g^b \mod p$
 (b geheim halten)

3. $A \xrightarrow{x=g^a} B$
 $B \xrightarrow{y=g^b} A$
 $A : y^a \mod p = g^{b \cdot a} \mod p = K$
 $B : x^b \mod p = g^{a \cdot b} \mod p = K$

2.5.9 Sicherheit

Angreifer: $p, g, g^a \mod p, g^b \mod p$

gesucht: $g^{a \cdot b} \mod p$

Einzig bekannte Möglichkeit ist das Berechnen a aus $g^a : (g^b)^a \mod p = K$ müsste diskretes Logarithmus-Problem lösen.

2.5.10 Man-in-the-Middle

M fängt g^a und g^b ab und wählt $c \in \{2, \dots, p\}$ und schickt $g^c \mod p$ an A und B .

$A : g^{c \cdot a} \mod p, B : g^{c \cdot b} \mod p$. Beide Schlüssel kennt auch M

2.6 ElGamal-Public Key Verfahren (1984)

2.6.1 Schlüsselerzeugung

A wählt p, g wie bei Diffie-Hellman. Wählt $a \in \{2, \dots, p-2\}, x = g^a \mod p$.

Öffentlicher Schlüssel: (p, g, x)

Geheimer Schlüssel: a

2.6.2 Verschlüsselung

Klartext $m : 1 \leq m \leq p-1$

$B \xrightarrow{m} A$

B wählt zufällig $b \in \{2, \dots, p-2\}$

$y = g^b \mod p$

Er berechnet $x^b \mod p$ und $f = m \cdot x^b \mod p$, sendet (y, f) an A ,

2.6.3 Entschlüsselung

$$y^a \bmod p (= x^b \bmod p)$$

$$\text{Berechnet } (y^a)^{-1} \bmod p [(y^a)^{-1} = (y^{\overbrace{p-1-a}^{\geq 0}})]$$
$$f \cdot (y^a)^{-1} \bmod p = m$$

Nachteil zu RSA

Doppelte Länge wird gebraucht, da Nachricht (Chiffre) und Teilschlüssel versendet werden.

2.7 Signaturen, Hashfunktionen, Authentifizierung

2.7.1 Anforderung an digitale Signaturen

Identitätseigenschaft: ID des Unterzeichners des Dokuments wird sichergestellt

Echtheitseigenschaft: des signiertem Dokument

Verifikationseigenschaft: Jeder Empfänger muss digitale Signatur verifizieren können.

2.7.2 RSA-Signatur (vereinfachte Version)

A will Dokument m signieren.

A besitzt öffentlichen RSA-Schlüssel (n, e) , geheimen Schlüssel d .

Signatur: $m^d \bmod n$ sendet $(m, m^d \bmod n)$ an B.

$$(m^d \bmod n)^e = m^{e \cdot d} \bmod n = m \bmod n$$

$$m < n$$

Wenn $m^{e \cdot d} \bmod n = m$, dann akzeptiert B die Signatur.

$m > n$ $m^d \bmod n$ B $\bmod n$. Ist $m' \bmod n = m \bmod n$, dann $(m', m^d \bmod n)$ gültige Signatur.

2.7.3 Wie lassen sich lange RSA-Signaturen vermeiden?

Def: Sei R ein endliches Alphabet.

Hashfunktion $H : \mathbb{R}^* \rightarrow R^k (k \in \mathbb{N} \text{ fest})$ soll effizient berechenbar sein.

2.7.4 RSA-Signatur mit HASH-Funktion

H öffentlich bekannte Hashfunktion.

A will Nachricht m signieren.

Bildet $H(m)$ und signiert $H(m) : H(m)^d \bmod n$ sendet $(m, H(m)^d \bmod n)$

Verifikation durch $B: m \rightarrow H(m)$
 $(H(m)^d \bmod n)^e \bmod n = H(m)$

2.7.5 Angriffsmöglichkeiten

- Angreifer kann $H(m)$ bestimmen wenn es ihm gelingt, $m' \neq m$ zu finden, so $(m', H(m)^d \bmod n)$ gültige Signatur von m durch A .
- Angreife wählt zufällig y und berechnet $y^e \bmod n = z$
 Gelingt es ihm, m zu finden mit $H(m) = z$, dann ist (m, y) gültige Signatur von m durch A
 $H(m) \cdot y^e = H(m)$

Def: Eine **kryptographische Hashfunktion** ist eine Hashfunktion, die folgende Bedingungen erfüllt.

1. H ist Einwegfunktion (um Angriffe des zweiten Typs zu vermeiden)
2. H ist **schwach kollisionsresistent**, d.h. zu gegebenem $m \in R^*$, soll es effizient nicht möglich sein ein $m' \neq m$, mit $H(m) = H(m')$, zu finden. (um Angriffe des ersten Typs zu vermeiden)

Verschärfung von 2.

- 2' H ist **stark kollisions resistent**, wenn es effizient nicht möglich ist $m \neq m'$ zu finden, mit $H(m) = H(m')$.

Da R^* unendlich und $|R^k| = |R|^k$ endlich ist, existiert unendlich viele Paare (m, m') , $m \neq m'$ mit $H(m) = H(m')$.

(Bilde $|R|^k + 1$ viele Hashwerte: Kollision)

Kollisionen lassen sich nicht vermeiden, sie sollten aber nicht schnell herstellbar sein.

2.7.6 Satz: Geburtstagsparadoxon

Ein Merkmal komme in m verschiedenen Ausprägungen vor. Jede Person besitze genau eine dieser Merkmalsausprägungen. Ist $c \geq \frac{1 + \sqrt{1 + 8 \cdot m \cdot \ln 2}}{2} \approx 1.18 \sqrt{m}$, so ist die Wahrscheinlichkeit, dass unter l Personen zwei die gleiche Merkmalsausprägung haben, mindestens $\frac{1}{2}$ (Geburtstage: $m = 366, l = 23$).

Beweis l Personen

Alle Möglichkeiten $(g_1, g_2, \dots, g_l), g_i \in \{1, \dots, m\}$ m^l Möglichkeiten.

Alle Merkmalausprägungen verschieden: $m \cdot (m-1) \cdot (m-2) \cdot \dots \cdot (m-(l-1))$

Wahrscheinlichkeit, dass l Personen lauter verschiedene Geburtstage haben.

$$q = \frac{m \cdot (m-1) \cdot (m-2) \cdot \dots \cdot (m-(l-1))}{m^l} = \prod_{i=0}^{l-1} 1 - \frac{i}{m}$$

Wann ist $q \leq \frac{1}{2}$?
 $e^x \geq 1 + x$

$$\begin{aligned} \prod_{i=0}^{l-1} 1 - \frac{i}{m} &\leq \prod_{i=0}^{l-1} e^{-\frac{i}{m}} = e^{\sum_{i=0}^{l-1} -\frac{i}{m}} \\ &= e^{-\frac{1}{m} \sum_{i=0}^{l-1} i} \\ &= e^{-\frac{1}{m} \cdot \frac{l(l-1)}{2}} \end{aligned}$$

$$\ln a \leq -\frac{1}{m} \cdot \frac{l \cdot (l-1)}{2} = -\frac{l^2 - l}{2 \cdot m}$$

2.7.7 Hashfunktion

$H(m) = H(m'), m \neq m'$

$H : \mathbb{Z}_2^* \rightarrow \mathbb{Z}_2^n$ (2^n Hashwerte)

Bei Erzeugung von circa $2^{\frac{n}{2}}$ Hashwerten ist die Wahrscheinlichkeit, dass zwei gleich sind ungefähr $\frac{1}{2}$.

$n = 64 : 2^{32}$ Hashwerte ($4 \cdot 10^9$) unsicher.

Weit verbreitet waren und sind:

MD5 (message digerst / Ron Rivest, 1991, 128 Bit)

SHA-1 (Secure Hash Algorithm, NSA, 1992/1993, 160 Bit)

2.7.8 Authentifizierung

Nachweise bzw. Überprüfung, dass jemand derjenige ist für den er sich ausgibt.
 Möglichkeiten der Authentifizierung durch:

Wissen

Besitz

biometrische Merkmale

gängigste Methode: Passwort

Im Allgemeinen: Passwort w abgespeichert als $f(w)$ f Einwegfunktion.

$w \xrightarrow{f^n} f^n(w) \xrightarrow{\text{sicher}} \text{Id. überprüfer } f$ Einweg.

1. Auth. $w_1 = f^{n-1}(w) \rightarrow f(f^{n-1}(w)) = w_0$ ersetzt w_0 durch w_1

2. Auth. $w_2 = f^{n-2}(w) \rightarrow \dots$

Passwortsicherheit: <http://www.schneier.com/crypto-gram-0701.html>

2.7.9 Challenge-Response-Authentifizierung

RSA-Verfahren $A \xrightarrow{\text{auth.}} B$

Öffentlicher Schlüssel: (n, e)

geheimer Schlüssel: d

$A \xleftarrow{\text{Zufallszahl } r} B, r < n \leftarrow \text{Challenge}$

$A \xrightarrow{r^d \bmod n} B$ überprüft, ob $r^{de} \bmod n = r \leftarrow \text{Response}$

Damit B sich sicher sein kann, dass es wirklich A ist, kann B so oft wie es für nötig hält neue r schicken und dadurch die Chance verringern, dass A nicht A ist.

2.8 Secret Sharing Scheme

Geheimnis wird auf mehrere Teilnehmer verteilt (Teilgeheimnisse), so dass gewisse Teilmengen der Teilnehmer das Geheimnis mit ihren Teilgeheimnissen rekonstruieren können, die anderen nicht.

$T = \{t_1, \dots, t_n\}, k < n$ (T Menge der Teilnehmer)

Jede Teilmenge von T mit mindestens k Teilnehmer sollen Geheimnis rekonstruieren können, Teilmengen von T mit weniger als k Teilnehmer nicht.

2.8.1 (k, n) - Schwellenwertsysteme

1979 Shamir (How to share a secret)

Konstruktion

Vereinbarung von großer Primzahl p , mindestens $p \geq n + 1$

$$g \in \mathbb{Z}_p = \{0, \dots, p-1\}$$

Verteilung der Teilgeheimnisse

Dealer wählt zufällig $a_1, \dots, a_{k-1} \in \mathbb{Z}_p, a_{k-1} \neq 0, k = \text{Schwelle}$

$$f(x) = g + a_1x + \dots + a_{k-1}x^{k-1} \in \mathbb{Z}_p[x]$$

(a_1, \dots, a_{k-1}) hält er geheim, natürlich auch g

Dealer wählt zufällig $x_1, \dots, x_n \in \mathbb{Z}_p$ (paarweise verschieden).

Teilnehmer t_i erhält als Teilgeheimnis $(x_i, f(x_i))$ (Punkt auf Polynom)

Bei $x = 0$ hast du g .

Rekonstruktion(sversuch) des Geheimnisses

k Teilnehmer $(x_{i_1}, f(x_{i_1})), \dots, (x_{i_k}, f(x_{i_k}))$

Durch diese Punkte ist f eindeutig bestimmt, z.B. durch Lagrange-Interpol.:

$$f(x_{i_j}) = g_{i_j}$$

$$f(x) = \sum_{j=1}^k g_{i_j} \cdot \frac{(x - x_{i_1}), \dots, (x - x_{i_{j-1}})(x - x_{i_{j+1}}), \dots, (x - x_{i_k})}{(x_{i_j} - x_{i_1}), \dots, (x_{i_j} - x_{i_{j-1}})(x_{i_j} - x_{i_{j+1}}), \dots, (x_{i_j} - x_{i_k})}$$

$$f(0) = g$$

$$g = \sum_{j=1}^k g_{i_j} \prod_{l \neq j} \frac{x_{i_l}}{(x_{i_l} - x_{i_j})}$$

Bei mehr als k Teilnehmer selbe Ergebnis.

Weniger als k Teilnehmer (k'): Anderes Polynom wegen weniger Punkte, also wahrscheinlich anderer g .

Erzeugen Polynom vom Grad $\leq k' - 1$

Für alle $k \in \mathbb{Z}_p$ existiert gleich viele Polynome vom Grad $\leq k' - 1$ durch die vorgegebene k' Punkte, die bei h durch y-Achse gehen.

Kapitel 3

Codierungstheorie

3.1 Grundbegriffe und einfache Beispiele

3.1.1 Codierung

(Kanalcodierung)

Sicherung von Daten/Nachrichten gegen zufällig auftretenden Fehler bei Speicherung/Übertragung.

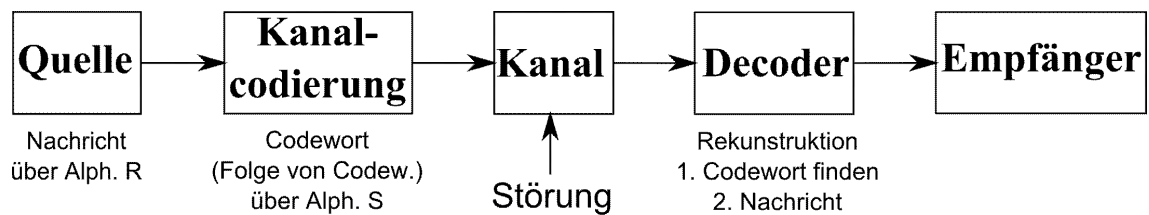


Abbildung 3.1: Schaubild der Codierung

3.1.2 Ziele

- Möglichst viele Fehler erkennen und gegebenenfalls korrigieren.
- Aufwand für Codierung und Decodierung möglichst gering.

3.2 Grundprinzip

Hinzufügen von Redundanz

Es gibt zwei Typen um Redundanz zu erzeugen.

3.2.1 FEC-Verfahren (Forward Error Correction)

Aufgetretene Fehler sollen erkannt und korrigiert werden.

Vorteil: keine Verzögerung der Übertragung aber ggf. große Redundanz notwendig.

3.2.2 ARQ-Verfahren (Automatic Repeat Request)

Aufgetretene Fehler sollen erkannt werden, werden nicht korrigiert. Stattdessen wiederholt die Übertragung beim Sender anfordern.

Vorteil: geringe Redundanz, aber Verzögerung.

Beispiele

3.2.3 Parity-Check-Codes

z.B. Nachrichten: 00, 01, 10, 11

Codierung:

00 → 000

01 → 011

10 → 101

11 → 110

(gerade Anzahl von Einsen in den Codewörtern)

1 Fehler wird erkannt, nicht korrigiert.

2 Fehler werden nicht erkannt.

3.2.4 Wiederholungscode

Nachrichten wie in 1.

Codierung:

00 → 000000

01 → 010101

10 → 101010

11 → 111111

(3-Fache Wiederholung)

1 Fehler wird erkannt und korrigiert.

$\underline{010101} \rightarrow 010101 \rightarrow 01$

Nachrichten wie in 1. Codierung:

$00 \rightarrow 00000$

$01 \rightarrow 01101$

$10 \rightarrow 10110$

$11 \rightarrow 11011$

Je zwei Codewörter unterscheiden sich an mindestens 3 Positionen.

Angenommen 1 Fehler tritt bei Übertragung auf. Dann gibt es genau ein Codewort, dass sich vom empfangenen Wort an genau einer Stelle unterscheidet; in das wird decodiert.

Muss immer Ungerade unterschiede in Codewörtern sein. Bei 5 diffs sind 2 Fehler korrigierbar.

3.2.5 (ehmaliger) ISBN-Code

International Standard Book Number

10-Stelliger Code

Erste 9 Ziffern haben inhaltliche Bedingung ($\hat{=}$ Nachricht)

10. Ziffer: Prüfziffer

Beispiel: 3-540-26121-? (Land - Verlag - Buchnummer - Prüfziffer)

Uncodierte Wörter sind gebildet über $R = \{0, \dots, 9\}$

Codierte Wörter sind gebildet über $S = \{0, \dots, 9, X\}$

ISBN-Wort $C_{10}C_9 \dots C_2C_1$

$C_{10} \dots C_2$ inhaltliche Bedingung, C_1 wird so gewählt, dass

$$\sum_{k=1}^{10} k \cdot C_k \equiv 0 \pmod{11}$$

$$10 \cdot C_{10} + \dots + 2 \cdot C_2 + C_1 \equiv 0 \pmod{11}$$

falls $C_1 = 10$ so setze $C_1 = X$

C_1 vom Beispiel ausrechnen.

$$10 \cdot 3 + 9 \cdot 5 + 8 \cdot 4 + 7 \cdot 0 + 6 \cdot 2 + 5 \cdot 6 + 4 \cdot 1 + 3 \cdot 2 + 2 \cdot 1 + C_1 \equiv 0 \pmod{11}$$

$$161 + C_1 \equiv 0 \pmod{11} \Rightarrow C_1 = 4$$

Ändern einer Ziffer wird erkannt:

$C_{10}C_9 \dots C_2C_1 \rightarrow C_i$ wird $X_i \neq C_i$ ersetzt

$$C_{10} \dots C_{i+1} X_i C_{i-1} \dots C_1$$

$$\sum_{k=1, k \neq i}^{10} k \cdot C_k + i \cdot x_i = \underbrace{\sum_{k=1, k \neq i}^{10} k \cdot C_k}_{\equiv 0 \pmod{11}} \cdot \overbrace{\left(\overset{\not\equiv 0 \pmod{11}}{i} \cdot \underbrace{(x_i - c_i)}_{\not\equiv 0 \pmod{11}} \right)}^{\not\equiv 0 \pmod{11}} \not\equiv 0 \pmod{11}$$

Fehler wird erkannt, Korrektur nicht möglich.

$$3 - 540 - 26121 - 4 \equiv 0 \pmod{11}$$

$$\left. \begin{array}{l} 3 - 540 - 26121 - \mathbf{6} \\ 3 - 540 - 261\mathbf{2}2 - 4 \end{array} \right\} \text{Prüfsumme 2.}$$

Vertauschung von Zwei Ziffern wird erkannt.

C_i und C_j vertauscht.

O.B.d.A $C_i \neq C_j$

$$C_{10} \dots C_j \dots \overset{\uparrow}{i} C_i \dots \overset{\uparrow}{j} C_1$$

$$\begin{aligned} \sum_{k=1, k \neq i, j}^{10} k \cdot C_k + i \cdot C_j + j \cdot C_i &= \sum_{k=1}^{10} k \cdot C_k + i(C_j - C_i) + j(C_i - C_j) \\ &= \underbrace{\sum_{k=1}^{10} k \cdot C_k}_{\equiv 0 \pmod{11}} + \underbrace{(C_j - C_i)}_{\not\equiv 0 \pmod{11}} \underbrace{(i - j)}_{\not\equiv 0 \pmod{11}} \not\equiv 0 \pmod{11} \end{aligned}$$

Vertauschung wird durch gewichtete Quersummen erkannt.

3.2.6 EAN-13-Code

European Article Number

13-Stelliger Code, erste 12 Ziffer sind inhaltlich festgelegt.

13. Ziffer ist Prüfziffer.

$$R = S = \{0, \dots, 9\}$$

$$C_1 \dots C_{12} C_{13}$$

$C_1 \dots C_{12}$ inhaltliche Angabe (in der Regel):

$C_1 C_2$ Herstellerland (40-43 Deutschland)

$C_6 \dots C_7$ Hersteller $C_8 \dots C_{12}$ interne Produktions Nummer

C_{13} so gewählt, dass

$$C_1 + 3 \cdot C_2 + C_3 + 3 \cdot C_4 + \dots + 3 \cdot C_{12} + C_{13} \equiv 0 \pmod{10}$$

$x \rightarrow 3x$ Permutation auf $\mathbb{Z}_{10} \pmod{10}$, da $ggT(3, 10) = 1$, 1 Fehler wird erkannt.

Vertauschung in der Regel nicht erkannt.

Übersetzung in Barcode:

$$C_1 C_2 \dots C_7 C_8 \dots C_{13}$$

Jede der Ziffern C_2, \dots, C_{13} wird durch einen 0-1-String der Länge 7 binär codiert.
 $0 \hat{=}$ weißer Balken, $1 \hat{=}$ schwarzer Balken.

Codierung sorgt dafür, dass nie mehr als 4 weiße oder schwarze Balken nebeneinander stehen.



Abbildung 3.2: EAN-13 Barcode

Schmalen Balken in Mitte und am Rand, sind nur Abtrennzeichen, die nichts mit EAN zu tun haben und nur beim einscannen helfen.

5 zu 0110001_2

C_2, \dots, C_7 werden nach Code A oder Code B codiert. C_1 bestimmt welcher dieser beiden Codes verwendet wird.

C_8, \dots, C_{13} werden nach Code C codiert.

C_1 ergibt sich aus der Art der Codierung von C_2, \dots, C_7

	Ziffern $C_2 - C_7$		Ziffern $C_8 - C_{13}$	bestimmt durch C_1
Zeichen	Code A	Code B	Code C	Code D
0	0001101	0100111	1110010	AAAAAA
1	0011001	0110011	1100110	AABABB
2	0010011	0011011	1101100	AABBAB
3	0111101	0100001	1000010	AABBBA
4	0100011	0011101	1011100	ABAABB
5	0110001	0111001	1001110	ABBAAB
6	0101111	0000101	1010000	ABBBAA
7	0111011	0010001	1000100	ABABAB
8	0110111	0001001	1001000	ABABBA
9	0001011	0010111	1110100	ABBABA

Codewörter von Code A,B oder C kommen nur einmal vor. Daher treten nie mehr als 4 gleiche Balken nebeneinander auf.

3.3 Blockcodes

00 → 00000
01 → 01101
10 → 10110
11 → 11011

3.3.1 Definition

S endl. Menge (=Alphabet), $n \in \mathbb{N}$.

Ein Blockcode C der (Block-)Länge n über S ist Teilmenge von $S^n = S \times \dots \times S$
 $\leftarrow n \rightarrow$

Elemente von C heißen **Codewörter**.

Ist $|S| = 2$ (i.d.R. $S = \{0, 1\}$), so **binär** Code.

$|C| = m$, so ist $m \leq |S|^n$.

Dann lassen sich n Informationssymbole (oder Strings von Informationssymbolen) codieren (Codierungsfunktion). Folge von Informationssymbolen (oder Strings) werden dann in Folge von Codewörtern codiert.

3.3.2 Definition: Hamming-Abstand

S endl. Alphabet, $n \in \mathbb{N}$.

$a, b \in S^n$ $a = (a_1, \dots, a_n)$, $b = (b_1, \dots, b_n)$

$d(a, b) = \#\{i : a_i \neq b_i\}$

Hamming-Abstand von a und b (Anzahl der unterschiedlichen Stellen).

(Richard W. Hamming, 1915-1998, Begründer der Codierungstheorie)

Eigenschaften

a) $d(a, b) = 0 \Leftrightarrow a = b$

b) $d(a, b) = d(b, a)$

c) $d(a, b) \leq d(a, c) + d(c, b)$ (Dreiecksungleichung)
 $(a_i \neq b_i \Rightarrow a_i \neq c_i \text{ oder } b_i \neq c_i)$

d) Wenn $(S, +)$ komm. Gruppe, dann auch S^n
 $[(a_1, \dots, a_n) + (b_1, \dots, b_n) = (a_1 + b_1, \dots, a_n + b_n)]$
 $d(a, b) = d(a + c, b + c)$ (Translationsinvarianz)

Also: Wird $x \in C$ gesendet und $y \in S^n$ wird empfangen und $d(x, y) = k$, so sind k Fehler aufgetreten.

3.3.3 Definition

a) Hamming-Decodierung

für Blockcode $C \subseteq S^n$

Wird $y \in S^n$ empfangen, so wird y zu einem Codewort $x' \in C$ decodiert, das unter allen Codewörtern minimalen Hamming-Abstand zu y hat.

$$d(x', y) = \min d(x, y), x \in C$$

(x' muss nicht eindeutig bestimmt sein)

z.B. $C = \{(0000), (1111)\}$

Empfangen: 0011 x' nicht eindeutig in diesem Fall.

($|S| = 2$: Hamming-Decodierung ist bestmöglich, falls jedes Symbol in einem Codewort mit der gleichen Wahrscheinlichkeit $p < \frac{1}{2}$ verändert wird und wenn jedes Codewort gleich wahrscheinlich ist.)

b) Minimalabstand

C Blockcode in S^n , Minimalabstand von C :

$$d(C) = \min d(x, x'), x, x' \in C, x \neq x'$$

(Ist $|C| = 1$, so $d(C) = n$)

[Bsp : $C = \{(00000), (01101), (10110), (11011)\}$, $d(C) = 3$]

c)

Ein Blockcode C ist **t-Felder-korrigierend**, falls $d(C) \geq 2t + 1$, und er heißt **t-Fehler-erkennend**, falls $d(C) \geq t + 1$.

Begründung für die Bezeichnung in c)

„Kugel“ vom Radius t um $x \in C$: $K_t(x) = \{y \in S^n : d(x, y) \leq t\}$

Ist $d(C) \geq 2t + 1$, so sind Kugeln vom Radius t um Codewörter disjunkt.

Angenommen es existiert $y \in S^n$ mit $y \in K_t(x) \cap K_t(x')$, $x, x' \in C, x \neq x'$. Dann $d(x, x') \leq d(x, y) + d(y, x') \leq t + t = 2t$. Widerspruch

$x \in C$ gesendet, y wird empfangen, und angenommen maximal t -Fehler sind aufgetreten, dann $y \in K_t(x)$ und Abstand zu jedem anderem Codewort ist $> t$

\Rightarrow Hamming-Decodierung ist korrekt.

$d(C) \geq t + 1$ und es treten maximal t minimal 1 Fehler auf, so ist y kein Codewort.

Bsp:

a) n -fach Wiederholungscode

$$S_n \rightarrow \underbrace{S_1 S_1 \dots S_1}_n$$

\vdots

$$S_k \rightarrow \underbrace{S_k S_k \dots S_k}_n$$

$$C = \{(s, s, \dots, s) : s \in S\} \subseteq S^n$$

$$d(C) = n$$

$$\left\lfloor \frac{n-1}{2} \right\rfloor \text{-Fehler-korr.}$$

b) ISBN, EAN-Codes, $d(C) = 2$, 1-Fehler-erkennend.

3.4 Titel???

$$d(C) \geq 2 \cdot t + 1, C \subseteq R^N$$

$$K_t(x) \cap K_t(x') = \emptyset$$

$$x, x' \in C, x \neq x'$$

y empfangen:

- falls y in $K_t(x)$ liegt für einen $x \in C$, so wird y nach x decodiert (Korrekt, falls max. t Fehler aufgetreten sind)
- falls y in keiner $K_t(x)$ liegt, so kann es mehrere Codewörter geben mit gleichem min. Abstand zu y . (Dann keine eindeutige Decodierung)

3.4.1 Definition: Perfekter Code

Code $C \subseteq R^n$ heißt perfekt, falls es ein $t \in \mathbb{N}_0$ gibt, mit der Eigenschaft:

$$R^n = \bigcup_{x \in C} K_t(x) \quad \text{und} \quad K_t(x) \cap K_t(x') = \emptyset \quad \text{für} \quad x, x' \in C, x \neq x'$$

Dann ist $d(C) = 2 \cdot t + 1$, falls $|C| > 1$:

Ang. $d(C) \leq 2 \cdot t$. Wähle $x, x' \in C, x \neq x'$, mit $d(x, x') = d(C) \leq 2 \cdot t$.

Wähle $y \in R^n$ mit $d(x, y) = t, d(y, x') \leq t$

$y \in K_t(x) \cap K_t(x')$ Widerspruch

$$d(C) \leq 2 \cdot t + 1$$

Wähle $x \in C$, wähle $y \in R^n$ mit $d(x, y) = t + 1$. Nach Voraussetzung existiert $x' \in C$ mit $y \in K_t(x')$.

$$d(x, x') \leq d(x, y) + d(y, x') \leq t + 1 + t = 2 \cdot t + 1$$

$$d(C) \leq 2 \cdot t + 1$$

3.4.2 Gibt es perfekte Codes?

Trivial Beispiele:

- einelementige Codes ($t=n$)
- $C = R^n$ ($t=0$)
(Jedes Element ist ein Codewort)
- n -fache Wiederholungscode über Z_2
 $n = 2 \cdot t + 1$
 $C = \{(0, \dots, 0), (1, \dots, 1)\}$
 $\quad \quad \quad \leftarrow n \rightarrow \quad \quad \leftarrow n \rightarrow$

3.4.3 Lemma

$|R| = q, x \in R^n, t \in \mathbb{N}$

Dann ist $|K_t(x)| = \sum_{i=0}^t \binom{n}{i} \cdot (q-1)^i$

$$\binom{n}{i} = \frac{n}{i(n-i)} \binom{n}{n-i}$$

Beweis

Abstand 0 zu x : 1 Word (nämlich x): $\binom{n}{0} \cdot (q-1)^0 = 1$

Abstand $i > 0$ zu x :

Anzahl der Auswahl von i Positionen aus n Positionen: $\binom{n}{i}$

An jeder Position $q-1$ Änderungsmöglichkeiten.

→ insgesamt $(q-1)^i$ Möglichkeiten,

Anzahl der Wörter vom Abstand i von x : $\binom{n}{i} \cdot (q-1)^i$

Satz

Sei C ein Code der Länge n über R , $|C| > 1, |R| = q$. Sei $t \in \mathbb{N}_0$ maximal mit $d(C) \geq 2 \cdot t + 1, t = \lfloor \frac{d(C)-1}{2} \rfloor$.

a) (Kugelpackungsschranke)

$$|C| \leq \frac{q^n}{\sum_{i=0}^t \binom{n}{i} \cdot (q-1)^i}$$

b) C ist perfekt \Leftrightarrow in a) gilt Gleichheit, d.h.

$$|C| = \frac{q^n}{\sum_{i=0}^t \binom{n}{i} \cdot (q-1)^i}$$

Beweis

a)

$d(C) \geq 2 \cdot t + 1$, daher $K_t(x) \cap K_t(x') = \emptyset$, $x \neq x'$, $x, x' \in C$

$$R^n \geq \bigcup_{x \in C} K_t(x)$$

$$q^n = |R^n|$$

$$\left| \bigcup_{x \in C} K_t(x) \right| = \sum_{x \in C} |K_t(x)| \stackrel{\text{Lemma}}{=} |C| \cdot \sum_{i=0}^t \binom{n}{i} \cdot (q-1)^i$$

b)

$$\Rightarrow: d(C) = 2 \cdot t + 1$$

$$R^n = \bigcup_{x \in C} K_t(x) \Rightarrow \text{Gleichheit in a)}$$

$$\Leftarrow: \text{Gleichheit} \Rightarrow R^n = \bigcup_{x \in C} K_t(x) \Rightarrow C \text{ perfekt.}$$

3.4.4 Bsp: Binärer Hamming-Code der Länge 7

$R = \mathbb{Z}_2 = \{0, 1\}$ C perfekt, $d(C) = 3$, $|C| = 16$

1-Fehler-Korrigierend

$$\begin{aligned} C = \{ (C_1, \dots, C_7) : C_i \in \mathbb{Z}_2, C_1 + C_4 + C_6 + C_7 = 0, \\ C_2 + C_4 + C_5 + C_7 = 0, \\ C_3 + C_5 + C_6 + C_7 = 0 \\ \} \subseteq \mathbb{Z}_2^7 \end{aligned}$$

C ist Unterraum von \mathbb{Z}_2^7

$$(C_1, \dots, C_7) \in C, (C'_1, \dots, C'_7) \in C$$

$$(C_1 + C'_1, \dots, C_7 + C'_7) \quad (C_1 + C'_1) + (C_4 + C'_4) + (C_6 + C'_6) + (C_7 + C'_7) = 0$$

$\dim(C) = 4$, C_4, C_5, C_6, C_7 frei wählbar $\curvearrowright C_1, C_2, C_3$ festgelegt

Basis:

$$(\dots 1000) \rightarrow (1101000)$$

$$(\dots 0100) \rightarrow (0110100) \quad |C| = 2^4 = 16$$

$$(\dots 0010) \rightarrow (1010010)$$

$$(\dots 0001) \rightarrow (1110001)$$

$d(C) = 3$:

Ang. $d(C) = d$. Wähle $x, x' \in C$ mit $d(x, x') = d$

Translationsinvarianz der Metrik:

$$\begin{aligned}
 d &= d(x, x') = d(x + x, x + x') = d(0, x + x') \\
 wt(x) &= \text{Anzahl der Einsen in } x \\
 &= d(0, x) \\
 d(C) &= \min wt(x), \quad x \in C, \quad x \neq \mathcal{V}
 \end{aligned}$$

Zeige: Jeder Vektor $\neq v$ in C enthält mind. 3 Einsen.

= 3 weist man nach durch überprüfen aller 15 von \mathcal{V} verschiedenen Codewörtern oder durch Analyse der Gleichung.

$$\begin{aligned}
 (C_1, \dots, C_7) &\in C \text{ Ang. } C_7 = 1 \\
 \Rightarrow C_1 + C_4 + C_6 &= 1. \text{ Wenn alle Eins } \checkmark \\
 C_1 = 1, C_4 = C_6 &= 0 \\
 C_4 = 1, C_1 = C_6 &= 0 \\
 C_6 = 1, C_1 = C_4 &= 0 \\
 C_1, C_2 \text{ oder } C_3 &= 1
 \end{aligned}$$

2. Fall: $C_7 = 1, C_4 = 1, C_1 = 0, C_2 \text{ oder } C_3 = 1$

3. Fall: analog zu Fall 2.

1. Fall: $C_1 = 1, C_4 = C_6 = 0, C_7 = 1$, o.B.d.A. $C_2 = C_3 = 0 \Rightarrow C_5 = 1$

$$d(C) \leq 3, \quad d(C) = 3 = 2 \cdot 1 + 1$$

Prüfe nach, ob bei Kugelpackungsschranke Gleichheit gilt:

$$\begin{aligned}
 |C| &= 16 \\
 |C| &\leq \frac{q^n}{\sum_{i=0}^t \binom{n}{i} \cdot (q-1)^i} \quad (q = 2, t = 1, n = 7) \\
 &= \frac{2^7}{1 + \binom{7}{1}} = \frac{2^7}{2^3} = 2^4 = 16 \\
 C &\text{ perfekt!}
 \end{aligned}$$

3.5 Lineare Codes

Def: Sei K ein endlicher Körper, $n \in \mathbb{N}$. Ein linearer Code C der Länge n ist ein Unterraum von K^n . (Zeilenvektoren) [Alphabet = k]

Ist $\dim(C) = k$, so heißt C $[n, k]$ -Code.

Ist $d(C) = d$, so $[n, k, d]$ -Code.

Beachte: $|K| = q \Rightarrow |C| = q^k$.

Informationsrate (Rate) von $C : \frac{k}{n}$.

3.5.1 Bemerkung über endliche Körper

- a) p Primzahl, \mathbb{Z}_p ist Körper der Ordnung p
- b) K endlicher Körper $\Rightarrow |K| = p^m$, p Primzahl, $m \in \mathbb{N}$.
- c) Zu jeder Primzahlpotenz p^m existiert (bis auf Isomorphie) genau ein Körper der Ordnung p^m .
- d) f sei irreduzibles Polynom vom Grad m über \mathbb{Z}_p . $K = \{g \in \mathbb{Z}_p[x] : \text{Grad}(g) \leq m-1\}$, $|K| = p^m$
 K wird Körper: Addition = übliche Addition von Polynomen
Multiplikation = normale Multiplikation + Reduktion mod f
(AES : $|K| = 2^8$)

3.5.2 Bsp

- a) n -facher Wiederholungscode über \mathbb{Z}_p
 $C = \{(0, \dots, 0), (1, \dots, 1), \dots, (p-1, \dots, p-1)\}$
 C ist linearer Code, $C = \langle (1, \dots, 1) \rangle$
 $[n, 1, n]$ -Code
- b) Hamming-Code ist linearer $[7, 4, 3]$ -Code über \mathbb{Z}_2
- c) $C = \{(c_1, \dots, c_n) : c_i \in \mathbb{Z}_p, \sum_{i=1}^n c_i = 0\}$
($p = 2$: Parity Check Code), linear $[n, n-1, 2]$ -Code über \mathbb{Z}_p
Basis von C : $(1, 0, \dots, 0, p-1), (0, 1, 0, \dots, 0, p-1), \dots, (0, \dots, 0, 1, p-1)$

3.5.3 Definition

K endl. Körper

- a) $x \in K^n$, so gewicht von x , $wt(x)$, definiert durch $wt(x) = \#\{i : x_i \neq 0\}$
- b) $\{0\} \neq C \subseteq K^n$, so ist das Minimalgewicht von C def. durch $wt(C) = \min_{x \in C, x \neq 0} wt(x)$

3.5.4 Satz

Ist $C \neq \{0\}$ ein linearer Code, so ist $d(C) = wt(C)$. (Beweis wie beim $[7, 4, 3]$ -Hamming Code)

3.5.5 Definition

Sei C ein $[n, k]$ -Code über K , sei $g_1 = (g_{11}, \dots, g_{1n}), \dots, (g_{k1}, \dots, g_{kn}) = (g_{k1}, \dots, g_{kn})$ eine Basis von C .

Dann heißt die $k \times n$ -Matrix $G = (g_1, \dots, g_k)^t = \begin{pmatrix} g_{11} & \dots & g_{1n} \\ \vdots & & \vdots \\ g_{k1} & \dots & g_{kn} \end{pmatrix}$ Erzeugermatrix von C

3.5.6 Satz

Sei G eine Erzeugermatrix von C . Dann ist $C = \{ u \cdot G : u \in K^k \}$ Beweis:

$$u = (u_1, \dots, u_k), u_i \in K$$

$$uG = (u_1, \dots, u_k) \cdot (g_1, \dots, g_k)^t = u_1 g_1 + \dots + u_k g_k \in C$$

3.5.7 Bemerkung

a) Die Abb $\{K^k \rightarrow C, u \mapsto uG$ ist bijektiv.

$u \in K^k$ Informationswörter

Codiert in Codewörter durch uG .

b) Elementare Zeilenumformungen an Erzeugermatrix liefern Erzeugermatrix.

3.5.8 Bsp: Hamming-[7, 4]-Code über \mathbb{Z}_7

Erzeugermatrix

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

Cod. eines Informationswort (u_1, u_2, u_3, u_4) ist G

$$(u_1, u_2, u_3, u_4) \rightarrow (u_1, u_2, u_3, u_4) \cdot G = (u_1, u_2, u_3, u_4, u_1+u_3+u_4, u_2+u_3+u_4, u_1+u_2+u_3)$$

3.5.9 Def

C $[n, k]$ -Code Erzeugermatrix C ist eine Standardform, falls ei folg. Gestalt hat.

$$\text{Cod. } (u_1, \dots, u_k) \cdot G = (u_1, \dots, u_k, *, \dots, *)$$

3.5.10 Satz

Sei C ein $[n, k]$ -Code über K . Dann existiert $(n - k) \times n$ -Matrix H über K mit folgenden Eigenschaften:

$$\text{Sei } y \in K^n. \text{ Dann: } y \in C \Leftrightarrow H \cdot y^t = \vec{0}$$

$$H \text{ heißt Kontrollmatrix von } C (\Leftrightarrow y \cdot H^t = \vec{0})$$

$$\text{Es ist } \text{rg}(H) = n - k \text{ (Dann ist } H \cdot G^t = 0)$$

3.5.11 Beweis

Sei g_1, \dots, g_k Basis von C , $G = (g_1, \dots, g_k)^t$

$g_i = (g_{i1}, \dots, g_{in})$

Betrachte LGS:

$$g_{11}x_1 + \dots + g_{1n}x_n = 0$$

$$\vdots$$

$$g_{k1}x_1 + \dots + g_{kn}x_n = 0$$

d.h. $G \cdot (x_1, \dots, x_n)^t = 0$. Koeffizientmatrix G hat Rang k .

Dim. des Lösungsraums dieses LGS = $n - k$

Sei $h_1, \dots, h_{n-k} \in K^n$ Basis des Lösungsraums dieses LGS.

$$H_i = (h_1, \dots, h_{n-k})^t, \quad H \cdot g^t = (h_1 g_i^t, \dots, h_{n-k} g_i^t) = 0, i = 1, \dots, k$$

$Hy^t = 0$ für alle $y \in C$.

$$rg(H) = n - k \Rightarrow \dim \text{Kern}(H) = k = \dim(C)$$

$$C = \text{Kern}(H)$$

3.5.12 Bemerkung

- Kontrollmatrix kann zur Fehlererkennung verwendet werden.
- Beweis liefert Verfahren: Erzeugermatrix \rightarrow Kontrollmatrix
- Umgekehrt: Kontrollmatrix \rightarrow Erzeugermatrix (Bilde Basis des Lösungsraums von $Hy^t = 0$)

3.5.13 Beispiel

a) Parity-Check-Code über \mathbb{Z}_p

$$C = \{(c_1, \dots, c_n) : \sum_{i=1}^n c_i = 0\}, \quad k = n - 1$$

$$H = (1, 1, \dots, 1)$$

$$H \cdot (c_1, \dots, c_n)^t = 0 \Leftrightarrow c_1 + \dots + c_n = 0 \Leftrightarrow (c_1, \dots, c_n) \in C$$

b) [7, 4]-Hamming-Code

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

c) C Code mit Erzeugermatrix

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} [4, 2]\text{-Code über } \mathbb{Z}_2$$

$$G \cdot (x_0, \dots, x_4) = 0$$

$$x_1 + x_2 + \quad \quad \quad x_4 = 0$$

$$x_2 + \quad \quad \quad x_4 = 0$$

x_5, x_4 frei wählen, x_1, x_2 festgelegt.

Basis (0010), (0101)

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

$$C\{(c_1, \dots, c_4) : c_3 = 0, c_2 + c_4 = 0\}$$