Codierung und Verschlüsselung

Steffen Lindner

January 26, 2016

Inhaltsverzeichnis

Kryptologie

1.1 Grundbegriffe und einfache Verfahren

Klartext: Unverschlüsselter Text (plain text)
Chiffre: Verschlüsselter Text (cipher text)
Sender Alice: Schlüssel $\rightarrow$ Verschlüsselung
Empfänger Bob: Schlüssel $\rightarrow$ Entschlüsselung

Encryption über Alphabet R, decryption über Alphabet S.
Verschlüsselung erfordert:

- Verschlüsselungsverfahren (Funktion E)
- Schlüssel k_e (encryption key) aus Menge K von möglichen Schlüsseln

E(m, k_e) = c, mit m Klartext, k_e Schlüssel und c Chiffre-text.
Entschlüsselung erfordert

- Entschlüsselungsfunktion (Funktion D)
- Schlüssel k_d (decryption key, hängt ab von k_e)

D(c, k_d) = m
Für festes k_e soll die Funktion E(., k_e) injektiv sein, d.h.

$$m_1 \neq m_2 \rightarrow E(m_1, k_e) \neq E(m_2, k_e) \tag{1.1}$$

D(., k_d) = E(., k_e)$^{-1}$

1.1.1 Definition

Ist $k_e = k_d$ (oder falls k_d einfach aus k_e zu berechnen ist), so spricht man von symmetrischen Verschlüsselungsverfahren.

Ist k_d nicht oder nur mit großem Aufwand aus k_e berechenbar, so spricht man von asymetrischen Verfahren.

Im zweiten Fall kann man k_e auch veröffentlichen: Public-Key-Verfahren.
Sicherheit eines Verschlüsselungsverfahrens darf nur von der Geheimhaltung von k_d abhängen. Bei symmetrischen Verfahren muss $k_e = k_d$ auf sicherem Wege ausgetauscht werden.

Asymmetrische Verfahren:
Bob erzeugt k_e, k_d.

1.1.2 Anzahl Schlüssel symetrisch vs. asymetrisch

Für symmetrische Verfahren gilt:

$$\binom{n}{2} = \frac{n \cdot (n-2)}{2} = O(n^2) \tag{1.2}$$

Für asymmetrische Verfahren gilt:

$$2n = O(n) \tag{1.3}$$

1.2 Beispiel

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

Verfahren: Verschiebechiffre

Menge der Schlüssel: K = $\{0, 1, ..., 25\}$

Elemente aus R werden einzeln verschlüsselt:

Wähle Schlüssel $i \in K$

Verschlüsseln: $x \in R \mapsto x + i$ mod 26

Entschlüsseln: $y \mapsto y - i$ mod 26

$m = x_1 ... x_r,\ x_j \in R$

$E(m, i) = ((x_1 + i) \bmod 26)\ ((x_2 + i) \bmod 26)\ ...\ ((x_r + i) \bmod 26) =$ c

$D(c, i) = (y_1 - i) \bmod 26)\ ...\ ((y_2 - i) \bmod 26) =$ m

Verfahren unsicher, da K klein.

(b) Verallgemeinerung: zeichenweise Substitutionschiffre

R = S = $\{0, ..., 25\}$

K = Menge aller Permutationen von R

Wähle Schlüssel $\pi \in K$, m = $x_1 ... x_r,\ x_j \in R$

Verschlüsseln:

$E(m, \pi) = \pi(x_1) ... \pi(x_r) = c$

Entschlüsseln:

$D(c, \pi) = \pi^{-1}(y_1) ... \pi^{-1}(y_r) = m$

$|K| = 26! \approx 4 \cdot 10^{26}$

Angenommen 10^{12} Schlüssel pro Sekunde testbar.

Angenommen 50% Schlüssel werden getestet.

Man benötigt dazu: $2 \cdot 10^{14}$ Sekunden ≈ 6.000.000 Jahre.

1.3 Übersprungen

1.4 Übersprungen

1.5 Prinzip von Kerkhoffs (1883)

Sicherheit eines Verschlüsselungsverfahrens darf nur von der Geheimhaltung des Schlüssels k_d abhängen, nicht von der Geheimhaltung des Verschlüsselungsalgorithmus (⇒ sollte offen gelegt werden: Praxistest).

1.6 Kryptoanalyse

Qualitative Unterschiede:

- Schlüssel k_d lässt sich ermitteln
- Ermittlung einer zu D(., k_d) äquivalenten Funktion (evtl. nur für gewisse k_e)
- Finden des Klartextes für einen speziellen Chiffretext

1.6.1 Arten von Angriffen

Ciphertext-only-Angriff

Lediglich der Chiffretext ist dem Angreifer bekannt.

Known-Plaintext-Angriff

Der Angreifer kennt bereits Plaintext - Chiffretext Kombinationen.

Chozen-Plaintext-Angriff

Der Angreifer kann sich mit einem selbstgewählten Plaintext Chiffretexte erzeugen.

Chozen-Ciphertext-Angriff

Der Angreifer kann sich gewünschte Chiffretexte entschlüsseln lassen.

One-Time-Pad und perfekte Sicherheit

2.1 Lauftextverschlüsselung

Früher im militärischen Bereich Lauftextverschlüsselungen.
Klartext: Endliche Folge z.B. über $\{0, ..., 25\}$.
Addiere (mod 26) anderen Text (z.B. ab gewisser Stelle im Buch).

Problem: Häufige Buchstaben treffen bei Addition häufig auf Buchstaben $\Rightarrow$ kryptoanalytische Möglichkeiten.

2.2 One-Time-Pad

Klartextalphabet R $= \mathbb{Z}_2 = \{0, 1\}$.

Verschlüsselung: Addiere zu Klartext der Länge n eine binäre Zufallsfolge der Länge n (Addition mod 2 = XOR).

Ergenibs: Chiffretext

$$\overbrace{m}^{Klartext} \oplus \overbrace{k}^{Zufallsfolge} = \text{c}$$

Entschlüsseln: $c \oplus k = m \oplus k \oplus k = m$

One-Time-Pad

k darf nur einmal verwendet werden.
$m_1 \oplus k = c_1$, $m_2 \oplus k = c_2$
$c_1 \oplus c_2 = m_1 \oplus k \oplus m_2 \oplus k = m_1 \oplus m_2$

2.3 Was ist Zufallsfolge der Länge n?

Frage ist sinnlos! Es kommt auf die Erzeugung an. Folge von Nullen und Einsen, die so erzeugt werden, dass jeses Bit unabhängig von den vorhergehenden mit Warscheinlichkeit $\frac{1}{2}$ erzeugt wird.
Jede Folge der Länge n hat die Wahrscheinlichkeit $\frac{1}{2^n}$.

2.4 One-Time-Pad ist perfekt sicher

Gegeben: Chiffretext c.

Dann gilt: $pr(m|c) = \overbrace{pr(m)}^{A-priori\ W-keit}$, für alle Klartexte m.

$pr(m|c) =$ Wahrscheinlichkeit, dass m Klartext war, wenn ich c kenne.

Substitutionschiffre ist nicht perfekt sicher:
Chiffretext: OCKTT
$\Rightarrow$ m = HAUCK kann nicht Klartext gewesen sein. $pr(m|c) = 0$

Symmetrische Blockchiffren

3.1 Blockchiffre

Klartext wird in Blöcke von fester Länge n zerlegt. Jeder Block wird einzeln verschlüsselt. Bei festem Schlüssel wird ein und derselbe Block immer in der gleichen Weise verschlüsselt (einfachstes Szenario).
Häufig: Klartextblocklänge = Chiffretextblocklänge

R = S = $\{1, 1\}$. Es gibt 2^n Blöcke der Länge n.
Jede Blockchiffre ist Permutation dieser 2^n Blöcke. Insgesamt $(2^n)!$ viele Blockchiffren der Länge n.
Schlüssel: Permutation
Speicherung: 2^n Bider von (0,...,0),...,(1,...,1). Insgesamt $n \cdot 2^n$ Bit.

Beispiel

n = 64. Ein Schlüssel erfordert 2^{70} Bit $\approx$ 13.5 Millionen Festplatten a 10 TB.

Lösung

Es wird nur eine ausgewählte Menge von Permutationen der Blöcke als Schlüsselmenge verwendet.

3.2 Lineare Chiffren

Klartextalphabet = Chiffretextalphabet = $\mathbb{Z}_k = \{0, ..., k-1\}$
Klartextblöcke: Elemente in $\mathbb{Z}_k^n$
Lineare Chiffre über $\mathbb{Z}_k$ (Blocklänge n):
m = $(V_1, ..., V_n)$, $V_i \in \mathbb{Z}_k$, Klartextblock.

Verschlüsseln

m $\underbrace{\rightarrow}_{Verschl.}$ $m \cdot A \in \mathbb{Z}_k^n$, A $n \times n$ - Matrix über $\mathbb{Z}_k$.

$$(V_1, ..., V_n) \cdot \begin{pmatrix} a_{11} & . & . & . & a_{1n} \\ . & & & & . \\ . & & & & . \\ . & & & & . \\ a_{n1} & . & . & . & a_{nn} \end{pmatrix} = (a_{11}V_1 + ... + a_{n1}V_n, ..., a_{1n}V_1 + ... + a_{nn}V_n)$$

Abb $m \to m \cdot A$ ist invertierbar (d.h. Permutation), wenn A invertierbar ist, das heißt wenn Matrix A^{-1} existiert mit $A \cdot A^{-1} = A^{-1} \cdot A = E_n$.
$c = m \cdot A$.

Entschlüsselung:
$c \cdot A^{-1} = m \cdot (A \cdot A^{-1}) = m \cdot E_n = m.$

Wann ist A invertierbar und wie berechnet man A^{-1}?
$\rightarrow$ Detrminante von A $det(A) \in \mathbb{Z}_k$.
$det(a) = a$
$det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = a_{11} \cdot a_{22} - a_{12} \cdot a_{21}.$

Inverse zu A (wenn sie existiert):

$$A^{-1} = \underbrace{(det(A))^{-1}}_{Inveres\ in\ \mathbb{Z}_k} \cdot B, B = (b_{ij}), b_{ij} = (-1)^{i+j} det(A_{ji}) \tag{3.1}$$

A_{ji} $(n-1) \times (n-1)$ - Matrix, die aus A durch streichen j-ten Zeile und i-ten Spalte entsteht.
det(A) ist invertierbar in $\mathbb{Z}_k \Leftrightarrow$ ggT(det(A),k) = 1

m $\rightarrow$ m $\cdot$ A ist invertierbar $\Leftrightarrow$ ggt(det(A),k) = 1.

Spezialfall n = 2

ggT$(a_{11} \cdot a_{22} - a_{12} \cdot a21, k) = 1$
$A^{-1} = (a_{11} \cdot a_{22} - a_{12} \cdot a_{21})^{-1} \cdot \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}$

3.2.1 Beispiel

$\mathbb{Z}_6$, k = 6, n = 2.

A = $\begin{pmatrix} 0 & 1 \\ 1 & 3 \end{pmatrix}$ det(A) = -1 mod 6 = 5

ggT(5,6) = 1

$A^{-1} = 5^{-1} \begin{pmatrix} 3 & 5 \\ 5 & 0 \end{pmatrix}$, $5^{-1} = 5$ in $\mathbb{Z}_6$.

$= 5 \cdot \begin{pmatrix} 3 & 5 \\ 5 & 0 \end{pmatrix}$.

Verschlüsseln von (1,3) mit A

(1,3) $\cdot \begin{pmatrix} 0 & 1 \\ 1 & 3 \end{pmatrix} \underbrace{=}_{mod\ 6} (3, 4).$

Nachricht (1 3 2 5)

(2,5) $\begin{pmatrix} 0 & 1 \\ 1 & 3 \end{pmatrix} = (5, 5).$

Chffretext: (3 4 5 5)

Entschlüsselung:

(3,4) $\cdot A^{-1} = (3,4) \cdot \begin{pmatrix} 3 & 1 \\ 1 & 0 \end{pmatrix}$ =(1 3)

(5.5) $\cdot A^{-1}$ = (2,5).

Anzahl der Schlüssel = Anzahl der invertierbaren $n \times n$ - Matrizen über $\mathbb{Z}_k$.
Für k = 2: $(2^n-1)(2^n-2)(2^n-2^2)...(2^n-2^{n-1})$
n = 64: $\approx 0,29 \cdot 2^{4096}$

3.3 Known-Plaintext-Angriff auf lineare Chiffren

$m \in \mathbb{Z}_k^n$, $m \to m \cdot A$, A invertierbare $n \times n$ - Matrix über $\mathbb{Z}_k$.
Angreifer kennt Klartextblöcke m_i und zugehörige Chiffretextblöccke c_i. Er will A ermitteln.
Er benötigt $\underline{n}$ Klartextblöcke $m_1, ..., m_n$ mit zugehörigen Chiffretextblöcken $c_1, ..., c_n$.
Daraus wird er häufig A ermitteln können:
$m_i \cdot A = c_i$, $i = 1, ..., n$.

M = $\begin{pmatrix} m_1 \\ m_2 \\ . \\ . \\ m_n \end{pmatrix}$, C = $\begin{pmatrix} c_1 \\ c_2 \\ . \\ . \\ c_n \end{pmatrix}$ $n \times n$ - Matrizen über $\mathbb{Z}_k$.

M · A = C
Falls M invertierbar ist, so berechne M^{-1}. Dann: $M^{-1} \cdot C = M^{-1}(MA) = (M^{-1}M)A = E_nA = A$.

Beispiel

n = 2, k = 26
Angneommen wir wissen: KRYPTO $\to$ QLIPRL mit linearer Chiffre.
$\underbrace{K}_{10}\underbrace{R}_{17}\underbrace{Y}_{24}\underbrace{P}_{15}\underbrace{T}_{19}\underbrace{O}_{14} \to \underbrace{Q}_{16}\underbrace{L}_{11}\underbrace{I}_{8}\underbrace{P}_{15}\underbrace{R}_{17}\underbrace{L}_{11}$
(n = 2)

M = $\begin{pmatrix} 10 & 17 \\ 24 & 15 \end{pmatrix}$, ggT(det(M), 26) = 1 ? Nein, da det(M) gerade.

Stattdessen:

M = $\begin{pmatrix} 10 & 17 \\ 19 & 14 \end{pmatrix}$, det(M) = $(10 \cdot 14 - 17 \cdot 19)$ mod 26 = 25.

$(det(M))^{-1}$ = -1 mod 26 = 25
$M^{-1} = (-1) \begin{pmatrix} 14 & -17 \\ -19 & 10 \end{pmatrix}$ mod 26

$= \begin{pmatrix} 12 & 17 \\ 19 & 16 \end{pmatrix}$.

Zugehöriges C: C = $\begin{pmatrix} 16 & 11 \\ 17 & 11 \end{pmatrix}$.

A = $M^{-1} \cdot C = \begin{pmatrix} 12 & 17 \\ 19 & 16 \end{pmatrix} \cdot \begin{pmatrix} 16 & 11 \\ 17 & 11 \end{pmatrix} \underbrace{=}_{mod\ 26} \begin{pmatrix} 13 & 7 \\ 4 & 21 \end{pmatrix}$

(10 17) A = (16 11)
(24 15) A = (8 15)
(19 14) A = (17 11)
$\rightarrow$ Test

3.4 Bemerkung

Häufig kann Sicherheit von Blockchiffren erhöht werden durch Hintereinanderausführung mehrerer Blockchiffren. Sinnlos bei linearen Blockchiffren:
2 x lineare Blockchiffre = 1x lineare Blockchiffre

3.5 Diffusion und Konfusion

Shannon, 1949 (Theory of Secrecy Systems)
Kriterien für gute Chiffrierverfahren.

3.5.1 Diffusion

Statistische Aufälligkeiten im Klartext sollen im Chiffretex nicht mehr erkennbar sein.
Insbesondere: Jedes Chiffretextzeichen muss von mehreren Klartextzeichen abhängen.

3.5.2 Konfusion

Aus statistischen Eigenschaften des Chiffretextes soll nicht auf den Schlüssel zurückgeschlossen werden.
Insbesondere: Jedes Chiffretextzeichen hängt von mehreren Schlüsselzeichen ab.

Lineare Chiffren: gute Diffusion, halbwegs gute Konfusion.

3.6 Feistel-Chiffren

Horst Feistel, IBM (1915-1990)
1971: Konstruktionsprinzip für symmetrische Blockchiffren. Wichtigste Realisierung: DES (Data Encryption Standard)

Prinzip:

1. Folge von Blocksubstitutionen und Blocktranspositionen (= Permutation der Einträge in einem Block). (Hohe Diffusion und Konfusion).

2. Erzeugung von Rundenschlüsseln aus Ausgangsschlüssel.

$m \in \mathbb{Z}_m^n$, n gerade

Klartextblock m

L_0 R_0

Bild kommt von Maxi
m = (L_0, R_0)
$L_i = R_{i+1}$
$R_i = L_{i-1} \oplus f_{k_i}(R_{i-1})$
$i = 1, ..., r-1$
$L_r = L_{r-1} \oplus f_{k_r}(R_{r-1})$
$R_r = R_{r-1}$
Konkrete Realisierung hängt von f ab + Art der Rundenschlüsselerzeugung.

Die Abbildungen $R_{i-1} \to f_{k_i}(R_{i-1})$ dürfen nicht affin-linear sein. (affin-linear: x $\mapsto xA + b$, Hintereinanderausführung affin-linearer Chiffren ist wieder affin-linear)
Denn sonst ist die gesamte Feistel-Chiffre affin-linear und angreifbar ähnlich wie lineare Chiffren.

Entschlüsselng einer Feistel-Chiffre = **Verschlüsselung** mit der umgekehrten Reihenfolge der Rundenschlüssel.

Der Advanced Encryption Standard (AES)

1970'er: Entwickeltdes DES (auf der Basis einer Feistelchiffre). Blocklänge n = 64, effektive Schlüssellänge 56 Bit. Anfällig gegen Brute-Force-Angriffe.

1997: Ausschreibung durch NIST für neuen Verschlüsselungsstandard

2000: Rijndael-Verfahren (J. Daemen, V. Rijmen)

2002: AES

4.1 Struktur des AES

AES ist iterierte Blockchiffre.
Blöcklängen: 128, 192, 256 Bit
Schlüssellängen: 128, 192, 256 Bit
Blocklängen unabhängig von Schlüssellängen
Anzahl der Runden (abhängig von gewählten Bitlängen), r= 10, 12, 14

Zwischenergebnisse nach jeder Runde:
Zustände (States), $S_0, ..., S_9$.
Jede Runde besteht aus 4 Transformationen:

$S_i \rightarrow$ Sub Bytes $\rightarrow$ Shift Rows $\rightarrow$ Mix Columns $\rightarrow \oplus K_{i+1} \rightarrow S_{i+1}$ (Bild)

Mix Columns fällt in der letzten Runde weg.
Vorbemerkung

128 Bit-Blöcke: 4×4 - Matrizen, jeder Eintrag Byte. $\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} \equiv$
Block $(a_{00}a_{10}a_{20}a_{30}a_{01}...a_{33})$ (Spaltenweise auslesen).

Bytes werden häufig als Elemente in einem Körper der Ordnung 2^8 aufgefasst (Anzahl der Elemente).

Beispiel für endlichen Körper: $\mathbb{Z}_p = \{0, ..., p-1\}$, p Primzahl

$$a \oplus b = a + b \mod p \tag{4.1}$$

$$a \odot b = a \cdot b \mod p \tag{4.2}$$

$\mathbb{Z}_2 = \{0, 1\}$, $\oplus$ XOR, $\odot$ $\cdot$, $\wedge$
Körper der Ordnung $2^8 : \mathbb{F}_{2^8}$
Elemente sind alle Polynome vom Grad < P über $\mathbb{Z}_2$:

$$a_7x^7 + ... + a_1x + a_0 \leftrightarrow (a_7, a_6, ..., a_0), \quad a_i \in \mathbb{Z}_2 \tag{4.3}$$

Addition in $\mathbb{F}_{2^8}$: Addition von Polynomen.
Multiplikation in $\mathbb{F}_{2^8}$: $f, g \in \mathbb{F}_{2^8}$

$$f \odot g := f \cdot g \mod h \tag{4.4}$$

wobei h das irreduzible Polynom $h = x^8 + x^4 + x^3 + x + 1$

Beispiel:
$(x^7 + x + 1) \odot (x^3 + x) = x^{10} + x^8 + x^4 + x^3 + x^2 + x \mod h = x^6 + x^5 + x^3 + 1$
In $\mathbb{F}_{2^8}$ existiert zu jedem $0 \neq g \in \mathbb{F}_{2^8}$ ein $g^{-1} \in \mathbb{F}_{2^8}$ mit $g \circ g^{-1} = 1$
Berechnung mit dem erweiterten Euklidischen Algorithmus (wie in $\mathbb{Z}$).

$$0 \neq f, g \in \mathbb{Z}_2[x] : u, v \in \mathbb{Z}_2[x] \text{ mit } u \cdot f + v \cdot g = ggT(f, g) \tag{4.5}$$

$0 \neq g \in \mathbb{F}_{2^8}, h$ irred. Polynom. ggT(g,h) = 1, da h irred.

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

(Verkürtzter) EEA in $\mathbb{Z}_2[x]$:
Input: $f, g \in \mathbb{Z}_2[x], g \neq 0$, Grad(f) $\geq$ Grad(g)
Output: $ggT(f, g), v \in \mathbb{Z}_2[x] : v \cdot g \mod f = ggT(f, g)$:

1. $s := f$, $t := g$, $v_1 := 0$, $v_2 := 1$, $v := 1$
2. Solange $s \mod t \neq 0$ wiederhole

   $q := s$ div $t, r := s \mod t$

   $v := v_1 - q \cdot v_2$, $v_1 := v_2$, $v_2 := v$

   $s := t, t := r$
3. Output: $t = ggT(f, g), v$

Zur Bestimmung von g^{-1} für $0 \neq g \in \mathbb{F}_{2^8}$ wende EEA an mit $f = h$. $g^{-1} = v \mod h$.

4.2 SubBytes-Transformation

Eingabe: Zustand $S_i = \begin{pmatrix} b_{00} & ... & b_{03} \\ . & & . \\ . & & . \\ b_{30} & ... & b_{33} \end{pmatrix}$, b_{ij} Bytes

Jedes Byte aus S_i wird einzeln verändert. Sei $g = (b_7, b_6, ..., b_0)$ ein Byte.

<u>1.Schritt:</u> $g \neq 0 : g \mapsto g^{-1}, 0 \mapsto 0$ (fasse g als Element in $\mathbb{F}_{2^8}$)
<u>2. Schritt:</u> Ergebnis $(c_7, c_6, ..., c_0)$ nach 1. Schritt wird affin-linearer Transformation unterworden:

$$(c_0, ..., c_7) \cdot A + b = (d_0, ..., d_7) \tag{4.6}$$

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

Übrige Zeilen durch zyklischen Shift der 1. Zeile um 1,2,...,7 Position nach rechts.

$b = (11000110)$
Ergebnis: $(d_7, ..., d_0)$
SubByes wird in AES durch Table Lookup in einer 16×16 - Matrix beschrieben: Einträge sind 0,...,255 (in gew. Reihenfolge).
Byte b $= (b_7, b_6, ..., b_0)$ bestimmt durch $b_7b_6b_5b_4$ die Zeile und $b_3b_2b_1b_0$ Spalte (Zeichen und Spalten sind mit 0,...,15 nummeriert). Eintrag an der entsprechenden Stelle ist Ergebnis von SubBytes angewendet auf b.

4.3 Shift Rows und MixColumns-Transformation

- Shift Rows: Jede der 4 Zeilen nach der 4×4 - Matrix (nach SubBytes) wird zyklisch nach links verschoben: i-te Zeile um i-1 Stellen (i=1,2,3,4).

- MixColumns: Elemente der Eingangsmatrix werden als Elemente in $\mathbb{F}_{2^8}$ betrachtet. Diese Matrix wird von links mit M = $\begin{pmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{pmatrix}$ (über $\mathbb{F}_{2^8}$) über $\mathbb{F}_{2^8}$ multipliziert.

  x entspricht (0, ...,0,1,0)

  1 entspricht (0, ...,0,1)

  $$\mathrm{M} \begin{pmatrix} a_{00} & . & . \\ a_{10} & . & . \\ a_{20} & . & . \\ a_{30} & . & . \end{pmatrix} = \begin{pmatrix} x \cdot a_{00} + x \cdot a_{10} + a_{10} + a_{20} + a_{30} & ... \\ a_{00} + xa_{10} + xa_{20} + a_{20} + a_{30} & ... \\ ... & ... \end{pmatrix}$$

4.4 Rundenschlüsselerzeugung

Ausgangsschlüssel hat 128 Bit. Wird geschrieben als 4×4 - Matrix von Bytes, spaltenweise zu lesen.
Spalten $w(0), w(1), w(2), w(3)$.
Definiere 40 weitere Spalten (a 4 Bytes).
w(0),...,w(i-1) seine schon definiert. $(i \geq 4)$
$i \not\equiv 0$ (mod 4): w(i) := w(i-4) $\oplus$ w(i-1) (Komponentenweise XOR-Verkn.)
$i \equiv 0$ (mod 4). w(i) := w(i-4) $\oplus$ T(w(i-1)), wobei T folgende Transformation ist:

$$w(i-1) = \begin{pmatrix} a & b & c & d \end{pmatrix}, \text{ a,b,c,d Bytes} \tag{4.7}$$

Wende auf b,c,d,a SubBytes an. Liefert e,f,g,h

Berechne:

$$r(i) = (00000010)^{\frac{i-4}{4}} \tag{4.8}$$

Potenz in $\mathbb{F}_{2^8}$.
$i = 4,\ x^0 = 1 \leftrightarrow (00000001)$
$i = 8,\ x^1 = x \leftrightarrow (00000010)$
$i = 12,\ x^2 \leftrightarrow (00000100)$
...
$i = 36$ x^8 in F_{2^8} reduziert mod $x^8+x^4+x^3+x+1 = x^4+x^3+x+1 \leftrightarrow (00011011)$

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

Rundenschlüssel K_i besteht aus Spalten $w(4i), ..., w(4i+3)$

4.5 Entschlüsselung

Alle Transformationen in AES sind invertierbar. mit der umgekehrten Reihenfolge der Rundenschlüssel Transformationen invertieren.

4.6 Schnelligkeit und Sicherheit

- Schnelligkeit:

  Software: 200 MBit/sec - 2 GBit/sec

  Hardware: 2GBit/sec - 70 GBit/sec

- Sicherheit:

  Nach 2 Runden vollständige Diffusion.

  Rundenschlüsselerzeugung mit SubBytes verhindert, dass Regelmäßigkeiten im Ausgangsschlüssel fortpflanzen. Verschiednee Ausgangsschlüssel haben nur sehr selten einen Rundenschlüssel gemeinsam.

  AES ist resistent gegen klassische kryptoanalytische Angriffe (differentielle Kryptoanalyse, lineare Kryptoanalyse).

Public-Key-Systeme

5.1 Grundidee (Diffie, Hellman, 1976)

Jeder Teilnehmer A hat Paar von Schlüsseln:

- P_A: öffentlicher Schlüssel
- G_A: geheimer Schlüssel

Zu jedem öffentlichen Schlüssel gehört öffentlich bekannte Verschlüsselungsfunktion E_{P_A} $(= E(., P_A))$.

B $\overbrace{\rightarrow}^{m}$ A. $m \rightarrow E_{P_A}(m) = c$ Chiffretext.

Bedingungen:

1) $E_{P_A}(m)$ muss schnell berechenbar sein, aber m darf für einen Angreifer aus Kenntnis von $E_{P_A}(m)$ nicht mit vertretbarem Aufwand berechenbar sein. (D.h. Invetierung von E_{P_A} ist schwierig)

   E_{P_A} ist <u>Einwegfunktion</u>.

2) A muss m aus c = $E_{P_A}(m)$ mit Hilfe von G_A effizient berechnen können:

$$m = D_{G_A}(c) \tag{5.1}$$

   Injektive Einwegfunktion, die mit Zusatzinformation leicht zu berechnen sind, heißen <u>Geheimtürfunktion</u> (trapdoor function).

   Aus 1) und 2) folgt:

3) G_A darf aus P_A nicht effizient berechenbar sein.

Es ist offenes Problem, ob Einwegfunktionen existieren.
Notwendig: $P \neq NP$.
Es gibt Kandidaten für Einwegfunktionen.

5.2 Das RSA-Verfahren

Rivest, Shamir, Adleman, 1977
Beruht auf der Schwierigkeit der Faktorisierung großer Zahlen.

(a) Schlüsselerzeugung

Wähle zwei große Primzahlen $p \neq q$ (z.B. ca 1000 Bit lang) (wie? später)

Bilde $n = p \cdot q$.

$\varphi(n) = |\{a : 1 \leq a \leq n, ggT(a,n) = 1\}| = (p-1) \cdot (q-1)$ (Eulersche φ-Funktion)

[Warum? Nicht teilerfremd zu n sind alle Vielfachen von p und alle Vielfachen von q, die $\leq$ n sind.

$1 \cdot p, 2 \cdot p, 3 \cdot p ..., (q-1) \cdot p$

$1 \cdot q, 2 \cdot q, 3 \cdot q, ..., (p-1) \cdot q,\ q \cdot p = \text{n}$

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

Teilerfremd: $n - q - p + 1 = pq - q - p + 1 = (p-1) \cdot (p-1)$]

Wähle $1 < e \leq \varphi(n)$ mit ggT(e, $\varphi(n)$) = 1 (Zufallszahl + Euklidischer Algorithmus).

Öffentlicher Schlüssel : $P_A = (n, e)$

Wähle $1 \leq d \leq \varphi(n)$ mit $ed \equiv 1 \pmod{\varphi(n)}$

[Wende erweiterten Euklidischen Algorithmus auf e und $\varphi(n)$ an. Liefert $s, t \in \mathbb{Z}$ mit $s \cdot e + t \cdot \varphi(n) = \text{ggT}(e, \varphi(n)) = 1$

$d := s \mod \varphi(n)$

$d \cdot e \mod \varphi(n) = (s \cdot e + \underbrace{t \cdot \varphi(n)}_{\equiv 0 \mod \varphi(n)}) \mod \varphi(n) = 1$]

Geheimer Schlüssel: $G_A = d$

(Jetzt kann man p, q und $\varphi(n)$ löschen, und sollte es auch.)

(b) Verschlüsselung

B $\overbrace{\to}^{Nachricht}$ A, B codiert Nachricht als Zahl. zerlege die Zahl in Blöcke, jeder der Blöcke sei als Zahl $< n$.

Sei m solch ein Block, aufjedenfall als Zahl.

Verschlüsseln von m: $m^e \mod n = c$

(c) Entschlüsselung

$c^d = (m^e)^d \mod n = m^{ed} \mod n = m$

Wieso gilt das?

Grund ist der kleine Satz von Fermat:

p Primzahl, $a \in \mathbb{Z}, p \not| a \Rightarrow a^{p-1} \mod p = 1$

[$a \mod p, a^2 \mod p, ...,\quad a^{n_1} \mod p = a^{n_2} \mod p, n_1 > n_2$

$a^{n_1 - n_2} \mod p = 1$, da $p \not| a$

Sei $m \in \mathbb{N}$ mit $a^m \mod p = 1$

U = $\{a \mod p, a^2 \mod p, ..., a^m \mod p\}$

$|U| = m$ U ist Untergruppe von $(\mathbb{Z}_p \setminus \{0\}, \odot)$

Satz von Lagrange: $|U| \mid |\mathbb{Z}_p \setminus \{0\}|, m|p-1$

p-1 = km

$a^{p-1} \mod p = a^{k \cdot m} \mod p = (a^m)^k \mod p = 1^k \mod p = 1.]$

ed = k $\cdot\varphi(n) + 1 = k \cdot (p-1) \cdot (q-i) + 1$

$m^{ed} \mod n = m \cdot m^{k \cdot (p-1) \cdot (q-1)} \mod n$

Ist $p \not| m$, so $m^{k(p-1)(q-1)} = (m^{p-1})^{k(q-1)} \mod p = 1$

$m^{ed} \mod p = m \cdot 1 \mod p = m \mod p$

Ist $p|m$, so $m \mod p = 0 = m^{ed} \mod p$

In jedem Fall: $m^{ed} \mod p = m \mod p$

Analog: $m^{ed} \mod q = m \mod q$

$\rightarrow m^{ed} \mod n = m \mod n \underbrace{=}_{m<n} m$

5.3 Berechnung modularer Potenzen

$m^e \mod n$

$$e = \sum_{i=0}^{k} e_i 2^i, \quad e_i \in \{0,1\}, e_k = 1$$

$m^e = m^{e_k 2^k + ... + e_1 2 + e_0} = m^{2^k} \cdot m^{e_{k-1} \cdot 2^{k-1}} \cdot ... \cdot m^{e_1 2} \cdot m^{e_0} = ((...((m^2 \cdot m^{e_k - 1})^2 \cdot m^{e_{k-1}})^2 ...)^2 m^{e_1})^2 \cdot m^{e_o}$
Suare-and-Multiply-Algorithmus $2 \cdot log_2(e)$ = Maximalzahl der erforderten Multiplikationen / Quadrierungen.

Nach jedem Rechenschritt mod n reduzieren. (In der Praxis will man Verschlüsseln beschleunigen: Man wählt e von der Form $2^a + 1$)

5.4 Sicherheit des RSA-Verfahrens

(a) Angreifer kennt n, nicht p,q (Faktorisierung ist schwer)

Wenn er $\varphi(n)$ kennt, so kann er d bestimmen (EEA)

$\varphi(n)$ aus n per Definition zu bestimmen ist aussischtslos.

Geht schnell, wenn er p,q kennt.

Tatsächlich: Fakt von n zu bestimmen bzw. $\varphi(n)$ zu bestimmen ist gleich schwierig.

Angenommen er kennt $\varphi(n)$.

$n = p \cdot q.$

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

Dann kennt er $p + q = ir$.

$n = p \cdot q = p(r-p) = -p^2 + pr$

Quadratische Gleichung für p (und für q). Lösung ist einfach.

Tatsächlich:

- Faktorisierung von n
- Berechnung von $\varphi(n)$
- Bestimmung von d

Gleichschwierige Probleme.

Offen: Ist die Bestimmung von m aus $c = m^e \mod n$ (bei Kenntnis von e und n) genauso schwierig wie Bestimmung von d?

(b) Berechnung von m aus $m^e \mod n = c$ ist einfach, falls $m^e < n$.

Dann: $m^e \mod n = m^e = c$

e-te Wurzel aus c: m

Einfach (binäre Suche).

Lösung für dieses Problem: Nur Teil von m ist tatsächlich Nachricht. Rest wird durch zufallsabhängiges (aber systematisches) Padding ergänzt.

Standard: OAEP (Optimal asymetric encryption padding)

(c) Brute Force für Faktorisierung.

Teste alle Zahlen $\leq \sqrt{n}$, ob sie Teiler von n sind.

Inputlänge: log(n)

$\sqrt{n} = 2^{\frac{1}{2} log(n)}$ exponentiell.

Unbekannt: Gibt es polyn. Faktorisierungsalgorithmen, d.h. mit Komplexität $O(log(n)^k)$, k fest?

Beste Faktorisierungsalgorithmen haben Komplexität $O(e^{c(log(n))^{frac13}(log(log(n)))^{\frac{2}{3}}})$, c $\approx 1,923$

Dezember 2009: 768-Bit-Zahl faktorisiert (Kleinjung), Dauer: 2000 CPU-Jahre (2GHz)

1024 Bit: c.a 1000 mal grßerer Aufwand.

Gilt nicht mehr als sicher.

2048 bit: 10^9 mal schwieriger als bei 1024-Bit Zahl

Vergleich: Brute Force für AES $\approx$ Faktorisierung von RSA - n mit 3064-Bit n.

5.5 Bestimmung großer Primzahlenn

Geht schnell.
Idee: Wenn p eine Primzahl, $1 \leq a \leq p-1$, so ist $a^{p-1} \mod p = 1$ (kleiner Satz von Fermat)
Sei $n \in \mathbb{N}$. Wähle $1 \leq a \leq n-1$. Teste, ob $a^{n-1} \mod n = 1$.
Wenn nein: n ist keine Primzahl!
Wenn ja: Wähle neues a. Fermat-Test
Wenn n Fermat-Test für alle $1 \leq a \leq n-1$ besteht, so ist n Primzahl.

Leider gibt es unendlich viele zusammengesetzte Zahlen n, die den Fermat-Test für "fast" alle a bestehen (nämlich für alle a mit ggT(a,n) = 1) (Carmichael-Zahlen)
Verbesserung: Miller-Rabin-test
Sei p eine Primzahl, $p \neq 2$. $p-1 = 2^e \cdot t$, $2 \nmid t$
$a^{p-1} = 1$ (Fermat)
$(a^{2^{e-1}\cdot t})^2 = 1$
$a^{2^{e-1}\cdot t}$ ist Nullstelle des Polynoms: $x^2 - 1 \in \mathbb{Z}_p[x]$
$\mathbb{Z}_p$ Körper $\Rightarrow x^2 - 1$ hat genau 2 Nullstellen in $\mathbb{Z}_p$, nämlich 1 und -1 $\mod p = p-1$
$a^{2^{e-1}\cdot t} \equiv \begin{cases} 1 \text{ (mod p)} \\ -1 \text{ (mod p)} \end{cases}$
Falls $e-1 \geq 1$ und falls $a^{2^{e-1}\cdot t} \equiv 1$ (mod p)
so $(a^{2^{e-2}\cdot t})^2 \equiv 1 \text{ (mod p)} \Rightarrow a^{2^{e-2}\cdot t} \equiv \begin{cases} 1 \text{ (mod p)} \\ -1 \text{ (mod p)} \end{cases}$
Ist p eine Primzahl, p-1 = $2^e \cdot t, 1 \leq a \leq p-1$, so gilt:
entweder $a^t \equiv 1$ (mod p) oder $a^{2^i\cdot t} \equiv -1$ (mod p) für ein $i \in \{0, ..., e-1\}$
Miller-Rabin: Teste mit n statt p.
Test nicht erfüllt: n ist keine Primzahl.
Test erfüllt: ? Wähle neues a!
Es gilt: Ist n keine Primzahl, so gibt es mindestens $\frac{3}{4}\varphi(n)$ a's, $1 \leq a \leq n-1$ und ggT(a,n) = 1, die zeigen, dass n keine Primzahl ist.
Besteht das n für mindestens $\frac{1}{4}\varphi(n)$ a's mit $1 \leq a \leq n-1$ und ggT(a,n) = 1 den Miller-Rabin-Test, so ist n Primzahl.

5.6 Das Diffie-Hellman-Verfahren zur Schlüsselvereinbarung (1976)

D-H ist kein Public-Key-Verfahren.
Schlüsselvereinbarung über unsicheren Kanal.
Es beruht auf folgendem Kandidaten für Einwegfunktion:

(a) p Primzahl, $\mathbb{Z}_p^* := \mathbb{Z}_p \setminus \{0\}$ ist Gruppe bezüglich Multiplikation $\odot$

Algebra: $\exists g \in \mathbb{Z}_p^*$ mit $\mathbb{Z}_p^* = \{g, g^2, ..., g^{p-1} = 1\}$

g heißt Primitivwurzel mod p.

$\{0, ..., p-2\} \rightarrow \{1, ..., p-1\}$

$a \mapsto g^a \mod p$

Kandidat für Einwegfunktion.

Umkehrfunktion: diskreter Logarithmus

Keine polyn. Algorithmen bekannt.

Beispiel: p = 7, g = 3

$\{3, 3^2 = 2, 3^3 = 6, 3^4 = 4, 3^5 = 5, 3^6 = 1\}$

(b) Das Verfahren:

A und B einigen sich auf große Primzahl p und Primitivwurzel g mod p.

Können öffentlich bekannt sein.

1. A wählt zufällig (geheimes) $a \in \{2, ..., p-2\}$ und berechnet $x := g^a \mod p$.

   A sendet x an B.
2. B wählt zufällig (geheimes) $b \in \{2, ..., p-2\}$ und berechnet $y := g^b \mod p$. B sendet y an A.
3. A berechnet $y^a \mod p = (g^b)^a \mod p = g^{ab} \mod p$
4. B berechnet $x^b \mod p = (g^a)^b \mod p = g^{ab} \mod p$

$K := g^{ab} \mod p$ ist gemeinsamer Schlüssel.

(c) Sicherheit:

Angreifer kennt $p, g, g^a \mod p, g^b \mod p$.

Möchte gerne $g^{ab} \mod p$ wissen!

D-H-Problem:

Gibt es schnellen Algorithmus zur Bestimmung von $g^{ab} \mod p$ bei Kenntnis von p,g $g^a \mod p, g^b \mod p$? - Unbekannt

Einziger bekannter Weg:

Berechne a aus x = $g^a \mod p$ (DLP = diskretes Logarithmus Problem) und dann K = $(y)^a \mod p$.

p sollte mindestens 2048 Bit haben.

5.7 Der Man-in-the-Middle-Angriff auf D-H-Verfahren

Mallory wählt zufälig $c \in \{2, ..., p-2\}$.

$$A \leftarrow M \rightarrow B$$

M fängt $x = g^a \mod p$ von A ab, $y = g^b \mod p$ von B ab.
M schickt A und B: $g^c \mod p$.
A berechnet $K_A = g^{ac} \mod p$ und glaubt, das ist der gemeinsame Schlüssel mit B.
B berechnet $K_B = g^{bc} \mod p$ und glaubt, das ist der gemeinsame Schlüssel mit A.

M kennt K_A und K_B.

$$A \overbrace{\longrightarrow}^{m} \underbrace{M}_{\text{entschlüsselt}} \quad \overbrace{\longrightarrow}^{\text{verschlüsselt mit } K_B} \quad B$$

Authentifizierung von A gegenüber B und umgekehrt notwndig.

5.8 ElGamal-Verschlüsselungsverfahren (T. ElGamal, 1984)

5.8.1 Schlüsselerzeugung:

A: p,g wie bei Diffie-Hellman.

a zufällig in $\{2, .., p-2\}, x = g^a \mod p$

Offentlicher Schlüssel: (p,g,x)

Geheimer Schlüssel: a

5.8.2 Verschlüsselung

Klartextblock m, $0 \leq m \leq p-1$, $B \longrightarrow A$
B wählt $b \in \{2, ..., p-2\}$ zufällig.
$y = g^b \mod p$ und $f := x^b \cdot m \mod p$

Er sendet (y, f) an A.

5.8.3 Entschlüsselung

A berechnet $y^a \mod p = x^b \mod p$ (= K)
$(y^a)^{-1} \cdot f = (x^b)^{-1} \cdot f = m$ (mod p)

Wie berechnet sie $(y^a)^{-1} \mod p$?

- Erweiterter Euklidischer Algorithmus (y^a, p)
- $(y^a)^{p-2} \mod p = (y^a)^{-1}$, denn $y^a \cdot (y^a)^{p-2} = (y^a)^{p-1} = 1$ (mod p) (Fermat)

Wichtig:
B muss bei Verschlüsselung von neuem Klartextblock neues b wählen, sonst kennt Angreifer $f_1 = x^b \cdot m_1$, $f_2 = x^b \cdot m_2$.

Angenommen Angreifer kennt m_1. Kennt $f_1 \cdot m^{-1} = x^b$ (mod p)
$(x^b)^{-1} \cdot f_2 = m_2$ (mod p)

Sicherheit von ElGamal = Sicherheit von Diffie-Hellman

5.9 Bug-Attacks

(Biham, Carmeli, Shamir) 2015 Journal of Cryptology

RSA-Verfahren (deterministisch) (RSA-CRT-Entschlüsselung).
$c = m^e \mod n$, $n = p \cdot q$.
$m_1 = c^d \mod p$
$m_2 = c^d \mod q$
$m_1, m_2 \longrightarrow m$ (Chinesischer Restsatz)

a, b 64-bit Zahlen.
$a \cdot b$ wir dauf 64-Bit-Mutliplier falsch berechnet. (1024-Bit n)

$$p < \sqrt{n} < q$$

Angreifer bildet C in der Nähe von $\sqrt{n}$.
In der Regel: $p < C < q$

(Chosen-Ciphertext-Angriff)
C entschlüsselt mit RSA-CRT.
$C \mod p$ enthält in der Regel nicht mehr a und b.
Entschlüsselung mod p $\rightarrow$ korrekt (M_1)
C mod q = C
$C^d \mod q$ berechnen $\rightarrow$ $\underline{a \cdot b}$ (BUG) wird auf Chip berechnet.
Entschlüsselung mod q nicht korrekt (M_2).
M_1, M_2 mit Chinesischem Restsatz zu M.
$M \mod p = M_1$
$M \mod q = M_2$

Angreifer erhält M: $M^e \mod n = C' \neq C$
$C' \mod p = C \mod p$
$C' \mod q \neq C \mod q$

$p|$ggT($C' - C$, n), $q \not| ggT(C' - C, n)$ $\rightarrow$ ggT$(C' - C, n)$ = p. RSA ist geknackt.

Signaturen, Hashfunktionen, Authentifizerung

6.1 Digitale Signaturen

Anforderung: Niemand außer A kann Dokument mit der Signatur von A versehen, selbst wenn er Signatur von A von anderen Dokumenten kennt.

Also auch: A kann nicht abstreiten, Dokument signiert zu haben.

Signatur gewährleistet:

- Identitätseigenschaft des Unterzeichners des Dokuments
- Echtheitseigenschaft des Dokuments

Außerdem: Verifikationseigenschaft

Jeder Empfänger einer von A signierten Nachricht muss Signatur verifizieren können.

Realisierung über RSA-Signatur.

6.2 RSA-Signatur

A will Dokument signieren. Sie besitzt öffentlichen RSA-Schlüssel (n,e), geheimen RSA-Schlüssel d.

Signatur von Dokument m ($m < n$):

$$\text{s(m)} := m^d \mod n$$

A sendet (m, $s_A(m)$) an B.
B verfiziert: $s_A(m)^e \mod n = m^{de} \mod n = m$, Signatur wird akzeptiert.

Falls $s_A(m)^e \mod n \neq m$, Signatur wird nicht akzeptiert.

Funktioniert so gut, da bei RSA Verschlüsselung / Entschlüsselung vertauschbar sind.

Signaturen sind auch mit ElGamal konstruierbar, komplizierter wegen Zufallswahl bei der Verschlüsselung.

Problem: Signatur ist so lang wie Nachricht.

Ausweg: Kryptografische Hashfunktionen.

6.3 Definition Hashfunktion

R endliches Alphabet. Hashfunktion ist Abbildung $R^* \to R^k$ (k fest)

6.4 RSA-Signatur mit Hashfunktion

A will m signieren (jetzt nicht notwendig $m < n$). Hashfunktion H der Länge k (Hashwerte $< 2^k \leq n$).
H öffentlich bekannt.
Sie sendet (m, $\underbrace{H(m)^d \mod n}_{s_A(m)}$)

Verifikation: B bildet H(m) aus m mit H. $s_A(m)^e \mod n = H(m)$?
$\to$ Ja ✓, Nein: nicht akzeptiert

6.5 Anforderung an H

(a) Angreifer kennt $(m, H(m)^d \mod n)$.

Er kann H(m) bestimmen. Gelingt es ihm ein $m' \neq m$ zu finden mit $H(m') = H(m)$, so ist $(m', H(m)^d \mod n)$ gültige Signatur von m' durch A.

(b) Angreifer wählt zufällig y und berechnet $y^e \mod n = z$. Gelingt es ihm ein m zu finden mit $H(m) = z$, so ist (m, y) eine gültige Signatur von m durch Alice.

Verifikation:

- $y^e \mod n = z$ ✓
- $H(m) = z$ ✓

6.6 Definition: Kryptographische Hashfunktion

Eine kryptographische Hashfunktion ist Hashfunktion H, die folgende Bedingungen erfüllen muss:

(1) H ist Einwegfunktion (um Angriffe vom Typ 6.5.b) zu unterbinden)

(2) H ist schwach kollisionsressistent, d.h. zu gegebenem n ist es nicht effizient möglich ein $m' \neq m$ zu finden mit $H(m') = H(m)$ (um Angriffe vom Typ 6.5.a) zu unterbinden) (second pre-image resistant)

Verschärftere Anforderung an H als (2):

(2') H ist stark kollisionsressistent, d.h. es ist nicht effizient möglich zwei $m_1 \neq m_2$ zu finden mit $H(m_1) = H(m_2)$

(Solche Kollisionen gibt es, sogar unendlich viele)

[Kryptographische Hashfunktion = message digest = digital fingerprint]

Wie lang sollte H(m) sein?

- Nicht zu lang (Sinn von Hashfunktion)
- Nicht zu kurz ("Geburtstagsparadox") $\rightarrow$ Geburtstagsattacke

(H(m) hat 128 Bit Länge. Angriff auf starke Kollisionsressistenz:
Wenn er $2^{128}+1$ H(m)'s erzeugt hat, hat er sicher Kollision)

6.7 Satz (Geburtstagsparadox)

Ein Merkmal komme in m verschiedenen Ausprägungen vor. Jedes Objekt (einer Grundgesamtheit) besitze genau eine dieser Merkmalesausprägungen (mit gleicher Wahrscheinlichkeit).

Ist dann $l \geq \frac{1+\sqrt{1+8\cdot m\cdot ln(2)}}{2}(\approx 1.18\sqrt{m})$, so ist die Wahrscheinlichkeit, dass unter l Objekten zwei die gleiche Merkmalsausprägung haben, $\geq \frac{1}{2}$.
(Geburtstagsparadoxon: m = 366, $l \geq 23 \rightarrow$ Wahrscheinlichkeit $\geq \frac{1}{2}$, dass 2 Personen am gleichen Tag Geburtstag haben)

6.7.1 Beweis

Gegeben l Objekte.
Alle Ereignisse: $(g_1, ..., g_l) \in \{1, 2, ..., m\}^l)$
Anzahl: m^l.
Alle g_i paarweise verschieden : $\prod_{i=0}^{l-1}(m-i)$

Wahrscheinlichkeit, dass keine 2 Objekte gleiche Merkmalsausspägung haben:

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

Benutze: $e^x \geq 1+x$.

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

Wann ist $q \leq \frac{1}{2}$?. Sicher, wenn
$e^{-\frac{l(l-1)}{2m}} \leq \frac{1}{2} \Leftrightarrow \frac{l(l-1)}{2m} \geq ln(2) \leftrightarrow l^2 - l - 2m\cdot ln(2) \geq 0 \Leftrightarrow l \geq \frac{1+\sqrt{1+8m\cdot ln(2)}}{2}$

6.8 Geburtstagsattacke (gegen starke Koll.res.)

Gegeben: H: $\{0,1\}^* \rightarrow \{0,1\}^k$
Angreifer erzeugt möglichst viele Hashwerte., diese werden auf Kollisionen untersucht.
Wende 6.7 an:

- Menge der Objekte = $\{0,1\}^*$
- Merkmalausprägungen = Hashwerte
- Anzahl aller möglichen Hashwerte m = 2^k

Wahrscheinlichkeit $\frac{1}{2}$, dass Kollision vorliegt, bei $\sqrt{2^k} = 2^{\frac{k}{2}}$ vielen erzeugten Hashwerten.

k = 64 keine ausreichende Sicherheit. Wahrscheinlichkeit $\frac{1}{2}$ Kollision bei 2^{32} $\approx 4 \cdot 10^9$ erzeugten Hashwerten

Forderung: $k \geq 128$, besser $k \geq 160$

6.9 Bemerkung

Weit verbreitete Hashfunktionen:

- MD5 (R. Rivest 1992) n = 128
- SHA-1 (NSA, NIST 1992/93/95) n = 160
- SHA-2-Familie (SHA-224, 256, 384, 512 NIST 2002/04)

Kollision bei SHA-1 mit ca. $2^{51} - 2^{57}$ Hashoperationen (vgl. mit 2^{80})

2007: NIST - Ausschreibung für neuen Stadard bei Hashfunktionen
2012: Gewinner KECCAK (Bertoni, Daemen, Peeters, van Assche). Länge des Hashwertes: 224 - 512 Bit
Konstruktionsprinzipien: Paar, Pelzl, Kap. 11

6.10 Authentifizierung

Nachweis / Überprüfung der Identität.
Forderung: Niemand anderes als A darf sich als A ausgeben können (auch nicht der Verifizierer).

Authentifizierung durch:

- Wissen
- Besitz
- biometrische Merkmale

6.11 Passwörter

A wählt Passwort w. Bei B, gegenüber dem sich A authentifizieren will, ist f(w) gespeichert, f Einwegfunktion (z.B. Hashfunktion).

→ Unsicher !

Besser: Einmal-Passwörter

Lamport, 1981:
f Einwegfunktion (öffentlich bekannt)
A: w_A, $f(w_A), f^2(w_A) = f(f(w_A)), ..., f^n(w_A)$
Am Anfang sendet A an B: $w_0 = f^n(w_A)$ (auf sicherem Weg)

Authentifizierung:

- A sendet $w_1 = f^{n-1}(w_A)$ an B
- B testet, ob $f(w_1) = w_0$.

Nach der ersten Authentifizierung ersetz B w_0 durch w_1. Nächste Authentifizierung $A \to B$ $f^{n-2}(w_A) = w_2$, B: $f(w_2) = w_1$. Etc.

Vorteil: Aus $f^i(w_A)$ lässt sich $f^{i-1}(w_A)$ nicht rekonstruieren.

6.12 Challenge-Response-Authentifizierungen

Signatur von Zufallsstring, z.B. mit RSA-Verfahren.
(n, e) öffentlicher Schlüssel von A, d geheimer Schlüssel von A)
$A \stackrel{auth.}{\to} B$
B wählt Zufallszahl $r < n$. A berechnet $r^d \mod n = c \to B$.
B verfiziert, ob $c^e \mod n = r$.

Wichtig: Authentifizierungsschlüssel und Verschlüsselungsschlüssel müssen verschieden sein !

Secret Sharing Schemes

Verteilung eines Geheimnisses auf mehrere Personen (Teilgeheimnisse), so dass gewisse Teilmengen dieser Personen mit ihren Teilgeheimnissen das Geheimnis rekonstruieren können, die anderen Teilmengen nicht.

7.1 Definition

(a) Sei T eine Menge von Teilnehmern, $\mathcal{Z} \subseteq \mathcal{P}(T)$ Menge der zulässigen Konstellationen. (Überlicherweise fordert man Monotonie von $\mathcal{Z}$:

$$A \subseteq C, A \in \mathcal{Z}, A \subseteq B \subseteq T \rightarrow B \in \mathcal{Z}$$

)

[T ={1,2,3,4}, $\mathcal{Z} = \{\{1,2\},\{1,3,4\},\{1,2,3\},\{1,2,4\},\{1,2,3,4\}\}$]

(b) Sei g ein Geheimnis (i.d.R. aus einer großen Menge möglicher Geheimnisse).

In einem Secret Sharing Scheme erhalten alle Teilnehmer durch eine vertrauenswürdige Instanz (Dealer) Elemente einer Menge $\mathcal{S}$ (Teilgeheimnisse, shadows), so dass gilt:

(1) Jede zulässige Konstellation kann mit ihren Teilgeheimnissen g rekonstruieren

(2) Unzulässige Konstellationen können das nicht

(c) SSS heißt perfekt, wenn jede unzulässige Konstellation mit ihren Teilgeheimnis genauso viel über g weiß wie ohne ihre Teilgeheimnisse.

7.2 Definition

T = $\{1, ..., n\}$, Sei k $\leq n$. Z = $\{U \subseteq T : |U| \geq k\}$

Ein SSS zu einem solchen Z heißt (k,n) - Schwellenwertsystem (threshold system)

7.3 Shamirs Konstruktion eines Schwellenwertsystems

(a) Vorgegeben sei große Primzahl p (in jedem Fall $p \geq n+1$).
$g \in \mathbb{Z}_p = \{0, ..., p-1\}$
Dealer wählt zufällig $a_1, ..., a_{k-1} \in \mathbb{Z}_p$, $a_{k-1} \neq 0$, er bildet

$$f(x) = g + a_1 x + ... + a_{k-1} x^{k-1} \in \mathbb{Z}_p[x]$$

($a_1, ..., a_{k-1}, g$ hält er geheim)

Dealer wählt paarweise verschiedene $x_1, ..., x_n \in \mathbb{Z}_p \setminus \{0\}$ (öffentlich bekannt).
Teilnehmer i erhält als Teilgeheimnis:

$$g_i = f(x_i) \text{ (und } x_i\text{)}$$

(b) Zur Rekonstruktion von g müssen k Teilnehmer kooperieren, etwa $i_1, ..., i_k$. Durch $(x_{i_1}, g_{i_1}), ..., (x_{i_k}, g_{i_k})$ ist das Polynom f eindeutig bestimmt.
Zum Beispiel mit LGS:

$$\begin{gathered} g + a_1 x_{i_1} + ... + a_{k-1} x_{i_1}^{k-1} = g_{i_1} \\ ... \\ g + a_1 x_{i_k} + ... + a_{k-1} x_{i_k}^{k-1} = g_{i_k} \end{gathered}$$

k Gleichungen mit k Unbekannten. Hat Lösung, und diese ist eindeutig (f, f' vom Grad $\leq k-1$, die an k Stellen den gleichen Wert haben, so hat $f - f'$ Grad $\leq k-1$ und mindestens k Nullstellen $\rightarrow f - f' = 0, f = f'$)
Oder sie bestimmen f durch sogenannte Interpolation (z.B. Lagrange-Interpolation).

$$f(x) = \sum_{j=1}^{k} \frac{(x - x_{i_1})...(x - x_{i_{-1}})(x - x_{i_{j+1}})...(x - x_{i_k})}{(x_{i_j} - x_{i_1})...(x_{i_j} - x_{i_{j-1}})(x_{i_j} - x_{i_{j+1}})...(x_{i_j} - x_{i_k})} \cdot g_{i_j} \quad (7.1)$$

Direkte Bestimmung von g durch Einsetzen von x = 0:

$$g = \sum_{j=1}^{k} g_{i_j} \cdot \prod_{l \neq j} \frac{x_{i_l}}{x_{i_l} - x_{i_j}} \quad (7.2)$$

Mehr als k Teilnehmer ezgeun dasselbe Polynom.

(c) Angenommen $k' < k$ Teilnehmer wollen g rekonstruieren. Für jedes $h \in \mathbb{Z}_p$ existieren gleich viele Polynome von Grad $\leq k-1$, die durch $(x_{i_1}, g_{i_1}), ..., (x_{i'_k}, g_{i'_k})$ und durch (0,h) gehen.

$\Rightarrow$ Shamir-Verfahren ist perfekt.

Codierungstheorie

8.1 Grundbegriffe und einfache Beispiele

8.1.1 Codierung (genauer Kanalcodierung)

Sicherung von Daten gegen Störungen bei der Übertragung / Speicherung.

Quelle (Nachricht über Alphabet R) $\longrightarrow$ Kanalcodierung (codiert Nachricht in Codewort oder Folge von Codewörtern über Alphabet S) $\longrightarrow$ Kanal (Störungen!) $\longrightarrow$ Decodierer (Rekonstruktion der Nachricht 1. Codewort zurückgewinnen 2. Nachricht zurückgewinnen) $\longrightarrow$ Empfänger

Ziel:

1. Möglichst viele Fehler erkennen und möglichst korrigieren
2. Aufwand für Codierung und Decodierung soll gering sein

Grundprinzip: Hinzufügen von <u>Redundanz</u> in systematischer Weise.

<u>Fehlererkennung</u> reicht, wenn Nachricht nochmal gesendet werden kann.

8.1.2 Beispiele

(a) <u>Parity-Check-Code</u>

R = $\{0,1\}$, $k \in \mathbb{N}$.

Nachricht wird in Blöcke der Länge k zerlegt.

Codierung: Block der Länge k $\to$ Block der Länge k+1

Anhängen eines Bits, so dass Anzahl der Einsen im Block der Länge k+1 gerade ist.

<u>k=2</u>

00 $\to$ 000

01 $\to$ 011

10 $\to$ 101

11 $\to$ 110

$\to$ 1 Fehler wird erkannt (kann nicht korrigiert werden), 2 Fehler werden nicht erkannt

(b) Wiederholungscode

Nachicht in Blöcke der Länge k zerlegen. Jeder Block wird m-mal wiederholt. (m-facher Wiederholungscode)

k=2, m=3

00 → 000000

01 → 010101

10 → 101010

11 → 111111

Wenn genau ein Fehler aufgetreten ist, kann er korrigiert werden.

Zum Beispiel:

001000 → 000000

Angenommen zwei Fehler sind aufgetreten. Es wird erkannt, dass Fehler aufgtreten ist.

Angenommen gesendet wurde 000000. Mögliche 2 Fehler:

- 000011 → 000000 ✓
- 000101 → 010101 falsch
- 001001 → ?

(c) Codiere Blöcke der Länge 2 über $\{0, 1\}$ folgendermaßen:

00 → 00000

01 → 01101

10 → 10110

11 → 11011

Je zwei Codewörter unterscheiden sich an mindestens 3 Stellen.

Wen genau ein Fehler aufgetreten ist und der Decodierer das "nächstgelegene" Codewort wöhlt, so Decoder korrekt.

Wenn 2 Fehler auftreten, wird erkannt, dass kein Codewort empfangen wird.

→ 1 Fehler korrigiert, 2 Fehler erkannt

8.1.3 GTIN-Prüfzifferncode (GTIN-13)

(a) GTIN: Global Trade Item Number (Artikelnr.)

13-stelliger Code: die ersten 12 Ziffern entsprechen Nachricht / Information, 13. Ziffer ist Prüfziffer

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

$c_1 ... c_{13}$

$\underline{c_1...c_{12}}$

Herstellungsland (i.d.R die ersten drei, D: 400-440)

Hersteller (i.d.R $c_4, ..., c_8$)

Produkt (i.d.R $c_9, ..., c_{12}$)

c_{13} so, dass $c_1 + 3_c s + c_3 + 3c_4 + ... + 3_c 12 + c_{13} \equiv 0 \ (mod 10)$

(b) GTIN wird übersetzt in Strichcode.

$1 \equiv$ schwarzer Balken

$0 \equiv$ weißer Balken

Ziffern $c_2, ..., c_7$ (linke Hälfte) werden nach Code A oder Code B binär codiert (mit jeweils 7 Stellen)

Blockcodes

Bei Codierung werden Wröter fester Länge n über Alphabet S gebildet, wobei nicht alle Wörter der Länge n Codewörter sind. (Wird ausgenutzt zur Erkennung/Korrektur von Fehlern)

9.1 Definition

S endliche Menge (Alphabet), $n \in \mathbb{N}$. Ein Blockcode $\mathcal{C}$ der (Block-)Länge n über S ist Teilmenge von $S \times S \times ... \times S =: S^n$
Elemente von $\mathcal{C}$: Codewörter
S = $\{0, 1\}$: binärer Blockcode.
Klar: $|\mathcal{C}| \leq |S^n|$.

Ist $|\mathcal{C}|$ = m, so lassen sich m Informationssymbole (oder Folgen von Onfosymbolen) - kurz Infoblöcke, den m Codewörtern zuordnen über eine Codierungsfunktion.

Folge von Infowörtern wird dann codiert in Folge von Codewörtern.

9.2 Definition

S endliches Alphabet, $n \in \mathbb{N}$
$a = (a_1, ..., a_n), b = (b_1, ..., b_n) \in S^n$
$d(a, b) = |\{i : 1 \leq i \leq n, a_i \neq b_i\}|$
(Hamming-)Abstand von a und b.

9.3 Bemerkung

(a) a,b,c $\in S^n$: d(a,c) $\leq$ d(a,b) + d(b,c)

(b) Ist S endliche Gruppe bezüglich + (kommutativ), so auch S^n und es gilt d(a,b) = d(a+c, b+c) (Translationsvarianz)

(c) Wird $x \in S^n$ gesendet und $y \in S^n$ empfangen und gilt d(x,y) = k, so sind k Fehler aufgetreten.

9.4 Definition

(a) Hamming-Decodierung für einen Blockcode $\mathcal{C} \subseteq S^n$:

Wird $y \in S^n$ empfangne, so wird y zu einem x' $\in \mathcal{C}$ decodiert mit d(x',y) = min (d(x,y)) $\forall x \in \mathcal{C}$ (x' muss nicht eindeutig sein!)

[$|S| = 2$: Hamming-Deociderung ist bestmöglich, wenn jedes Symbol unabhängig von dem anderen mit Wahrscheinlichkeit $p < \frac{1}{2}$ verändert wird und wenn alle Codewörter gleich wahrscheinlich sind.

Dann ist die Hamming-Decodierung die maximum-likehood-Decodierung, d.h. sie maximiert p(x|y), x $\in \mathcal{C}$, $y \in S^n$]

(b) $\mathcal{C}$ Blockcode in $S^n, |\mathcal{C}| > 1$.

Minimalabstand von $\mathcal{C}$ ist d($\mathcal{C}$) := mind(x,x')), $x, x' \in \mathcal{C}, x \neq x'$

(c) Ein Blockcode $\mathcal{C}$ heißt t-Fehler-korrigierend, falls d($\mathcal{C}$) $\geq$ 2t+1 und t-Fehler-erkennend, falls $d(\mathcal{C}) \geq t + 1$

9.5 Bemerkung

(a) Ist $d(\mathcal{C}) \geq 2t + 1$, so sidn die "Kugeln" vom Radis t (bzgl. Hamming-Abstand) um Codewörter x als Mittelpunkt disjunkt.

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

[$x, x' \in \mathcal{C}, x \neq x', y \in K_t(x) \cap K_t(x')$, so $d(x, x') \leq d(x, y) + d(y, x') \leq t + t = 2t$

Widerspruch zu $d(\mathcal{C}) \geq 2t + 1$

Sind also bei Übertragung eines Codewortes höchstens t Fehler aufgetreten, so ist Hamming-Decodierung korrekt]

(b) Ist $d(\mathcal{C}) \geq t + 1$ und treten bei der Übertraung von x maximal t Fehler auf (und mind. einer), so ist das empfangene Wort μ kein Codewort: $1 \leq d(x, y) \leq t$

D.h. es wird bemerkt, dass Fehler aufgetreten sind.

9.6 Beispiel

(a) GTIN, Parity-Check-Code 1 Fehler erkennend

(b) m-facher Wiederholungscod $d(\mathcal{C}) = n, \mathcal{C}$ ist $\frac{m-1}{2}$ Fehler korrigierend. $\mathcal{C}$ Code mit $d(\mathcal{C}) = 2t + 1$, $K_t(x) \cap K_t(x') = \emptyset$ für alle $x, x' \in \mathcal{C}, x \neq x'$

Liegt empfangenes Wort in einer dieser Kugeln, so Hamming-Decodierungen eindeutig (und korrekt falls max. t Fehler aufgetreten sind).

Liegt ein Wort in keiner dieser Kugeln, so kann es Mehrdeutigkeiten bei Decodierung geben. Besonders schöne Situation: jedes Wort liegt in einer dieser Kugeln.

9.7 Definition

Sei $\mathcal{C}$ Blockcode über $\mathcal{C}$ der Länge n. $\mathcal{C}$ heißt perfekt, falls es $t \in \mathbb{N}_0$ gibt mit $\mathcal{C}^n = \bigcup_{x \in \mathcal{C}} K_t(x)$ (Disjunkt).

9.8 Bemerkung

Ist $\mathcal{C}$ perfekt $|\mathcal{C}| > 1$, t wie in 9.7, so ist $d(\mathcal{C}) = 2t + 1$.

9.8.1 Beweis

Klar $d(\mathcal{C}) > 2t$. Angenommen $d(\mathcal{C}) = |\mathcal{C}|$
Wähle $y \in \mathcal{C}^n$ mit $d(x, y) = t$ und $d(x)$
In orginalscript nachschauen.

9.9 Lemma

$|S| = q, x \in S^n, t \in \mathbb{N}$. Dann ist $|K_t(x)| = \sum_{i=0}^{t} \binom{n}{i}(q-1)^i$

9.9.1 Beweis

Abstand i zu x: i Positionen von x auswählen $\binom{n}{i}$ Möglichkeiten. An jeder Position Eintrag von x ändern $(q-1)^i$ Möglichkeiten. Also $\binom{n}{i}(q-1)i$ Wörter $y \in S^n$ mit $d(x, y) = i$.

9.10 Satz

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

(a) (Hamming-Schranke, Kugelpackungsschranke)

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

(b) $\mathcal{C}$ perfekt $\Leftrightarrow$ Gleichheit bei (a), d.h.

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

9.11 Beispiel

Zu 8.2.c).
$|\mathcal{C}| = 4$. $d(\mathcal{C}) = 3, n = 9$ über $\mathbb{Z}_2$.
Geht das auch mit $n = 4$?
$3 = 2 \cdot 1 + 1 \Rightarrow t = 1$.

$$4 \leq \frac{2^4}{1+4} < 4$$

Widerspruch!
$\Rightarrow$ einen solchen Code gibt es nicht.

9.12 Beispiel

(a) einelementige Codes (t=n) sind perfekt.

(b) $\mathcal{C} = S^n$ (t=0) ist perfekt.

(c) n-facher Wiederholungs-Code über $\mathbb{Z}_2$

(d) Nicht triviales Beispiel für perfekten Code: S = $\mathcal{Z}_2$

$\mathcal{C} = \{(c_1, ..., c_7) \in \mathbb{Z}_2^7, c_1 + c_4 + c_6 + c_7 = 0,\ c_2 + c_4 + c_5 + c_7 = 0,\ c_3 + c_5 + c_6 + c_7 = 0\}$ (Rechnen in $\mathcal{Z}_2$)

$\mathcal{C}$ ist perfekt, $|\mathcal{C}| = 2^4, d(\mathcal{C}) = 3$.

Das ist einfaches Beispiele einer Serie von perfekten Codes, den Hamming-Codes, die wir später noch behandeln werden.

Linear Codes

10.1 Definition

Sei K ein Körper, $n \in \mathbb{N}$. Ein linearer Code $\mathcal{C}$ ist ein Unterraum des K-VR $\in K^n$

Beachte $|K| = q$, so $|\mathcal{C}| = q^k$.

$\frac{K}{n}$ heißt Informationsrate des Codes.

10.2 Bemerkung zu endl. Körpern

(a) $\mathbb{Z}_n$ ist bezgl. Additiun und Mutliplikation *mod* n ein endl. Körper, falls n Primzahl

10.3 Beispiel

(a) n-facher Wiederholungscode über $\mathbb{Z}_p$

$\mathcal{C} = \{0, ..., 0), (1, ..., 1), ..., (p-1, ..., p-1)\}$ ist ein linearer Code, [n,1,n]-Code

(b) Hamming-Code ist linearer [7,4,3]-Code über $\mathbb{Z}_2$.

(c) $\mathcal{C} = \{(c_1, ..., c_n) : c_i \in \mathbb{Z}_p, \sum_{i=1}^{n} c_i = 0\}$ linearer [n,n-1,2]-Code über $\mathbb{Z}_p$ (p=2 $\rightarrow$ Parity-Check-Code)

10.4 Definition endl. Körper

(a) $x \in K^n$, so Gewicht von $x = (x_1, ..., x_n), wt(x)$ definiert durch $wt(x) = |\{i : 1 \leq i \leq n, x_i \neq 0\}|$

(b) Ist $\{0\} \neq \mathcal{C} \subseteq K^n$, so ist das Minimalgewicht von $\mathcal{C}$ definiert durch $wt(\mathcal{C}) = min\ wt(x)$

10.5 Satz

Ist $\mathcal{C} \neq \{0\}$ ein linearer Code, so ist $d(\mathcal{C}) = wt(\mathcal{C})$

10.6 Definition

Sei $\mathcal{C}$ ein [n,k]-Code über K und $g_1 = (g_{11}, ..., g_{1n}), ..., g_k = (g_{k1}, ..., g_{kn})$ eine Basis von $\mathcal{C}$.
Dann heißt die $k \times n$ - Matrix

$$G = \begin{pmatrix} g_1 \\ ... \\ g_k \end{pmatrix} = \begin{pmatrix} g_{11} & ... & g_{1n} \\ ... & ... & ... \\ g_{k1} & ... & g_{kn} \end{pmatrix}$$

Erzeugermatrix von $\mathcal{C}$.

10.7 Satz

G Erzeugermatrix eines [n,k]-Codes $\mathcal{C}$ über K, so

$$C = \{\underbrace{u}_{1\times k} \cdot \underbrace{G}_{k\times n} : u \in K^k\}$$

10.7.1 Beweis

$$u \cdot G = (u_1, ..., u_k) \cdot \begin{pmatrix} g_1 \\ ... \\ g_k \end{pmatrix} = u_1 g_1 + ... + u_k g_k$$

Auf diese Weise entstehen alle Codewörter.

10.8 Bemerkung

(a) Die Abbildung $\begin{cases} K^k \to \mathcal{C} \subseteq K^n \\ u \mapsto u \cdot G \end{cases}$ ist injektiv.

Damit: Codierungsmöglichkeit von Informationswörtern der Länge k in Codewörter der Länge n.

(b) Elementare Zeilenumformung an Erzeugermatrix von $\mathcal{C}$ liefern wieder Erzeugermatrix von $\mathcal{C}$

10.9 Beispiel

Hamming-Code [7,4]-Code über $\mathbb{Z}_2$ (9.12.d).

$\mathcal{C} = \{(c_1, ..., c_7) \in \mathbb{Z}_2^7 : c_1+c_4+c_6+c_z = 0, c_2+c_4+c_5+c_7 = 0, c_3+c_5+c_6+c_7 = 0\}$

$c_4, ..., c_7$ frei wählbar, dann c_1, c_2, c_3 bestimmt.

Erzeugermatrix:

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

Durch elementare Zeilenumformungen andere Erzeugermatrix:

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

Codierung von $u = (u_1, u_2, u_3, u_4)$ mit $\tilde{G} : u/cdot\tilde{G} = (\underbrace{u_1, u_2, u_3, u_4,}_{\text{Information}} *, *, *)$

(Eine Erzeugermatrix von der Form $(E_n *)$ heißt in Standardform. Nicht jeder Code besitzt erzeugermatrix in Standardform.)

10.10 Satz und Definition

Sei $\mathcal{C}$ ein [n,k]-Code über K. Dann existiert $(n-k) \times n$ - Matrix H, sodass gilt:

$$\text{Ist } y \in K^n \text{, so: } y \in \mathcal{C} \Leftrightarrow H \cdot y^t = \sigma \ (\Leftrightarrow y \cdot H^t = 0)$$

H heißt Kontrollmatrix von $\mathcal{C}$. Es ist $rg(H) = n - k$.
(Dann gilt auch: $H \cdot G^t = 0$ - Nullmatrix)

10.10.1 Beweis

Sei $g_1, ..., g_k$ Basis von $\mathcal{C}$. $G = \left(g_1, ..., g_k\right), g_i = (g_{i_1}, ..., g_{i_n})$

Betrachte homogenes LGS: $G \cdot \begin{pmatrix} x_1 \\ ... \\ x_n \end{pmatrix} = 0$, bzw. $(x_1, ..., x_n) \cdot G^t = 0$ (Zeilenvektor der Länge k)
Zeilen von G sind lin. unabhängig, d.h. $rg(G) = k$.

Daher: Dimension des Lösungsraums von $G \cdot \begin{pmatrix} x_1 \\ ... \\ x_n \end{pmatrix} = 0$ ist $n - k$.

$h_1 = \begin{pmatrix} h_{11} \\ ... \\ h_{1n} \end{pmatrix}, ..., h_{n-k} = \begin{pmatrix} h_{n-k,1} \\ ... \\ h_{n-k,n} \end{pmatrix}$ Basis des Unterraums.

$H = \begin{pmatrix} h_1^t \\ ... \\ h_{n-k}^t \end{pmatrix}$, $G \cdot h_1 = 0, ..., G \cdot h_{n-k} = 0$.

Dann gilt:

- $g_j \cdot h_i = 0, \forall i, j$
- $g_j \cdot H^t = 0, \forall j$

- $H \cdot y^t = 0, \forall y \in \mathcal{C}$

(*) $\mathcal{C} \subseteq$ Lösungsraum von $H \cdot y^t = 0$
Dimension Lösungsraum von $H \cdot y^t = 0$ ist $n - rg(H) = n - (n-k) = k$.

Daher gilt (*).

10.11 Bemerkung

(a) Kontrollmatrix kann zur Fehlererkennunng verwendet werden.

(b) Beweis von 10.10 $\rightarrow$ Verfahren zur Bestimmung von H aus G.

(c) Geg. H. Bestimme Basis des Lösungsraums von $H \cdot y^t = 0 \rightarrow G$

10.12 Beispiel

(a) Parity-Check-Code über $\mathbb{Z}_2$, $\mathcal{C} = \{(u_1, ..., u_n) : \sum u_i = 0 \text{ in } \mathbb{Z}_2\}$

Kontrollmatrix: $(1, ..., 1) \cdot \begin{pmatrix} u_1 \\ ... \\ u_n \end{pmatrix} = u_1 + ... + u_n = 0$

(b) Hamming [7,4]-Cde aus 10.9:

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

Kontrollmatrix von $\mathcal{C}$.

(c) $\mathcal{C}$ Code mit Erzeugermatrix $\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$ [4,2]-Code über $\mathbb{Z}_2$.

Gesucht: Kontrollmatrix

Es muss gelten:

$$\begin{aligned} x_1 + x_4 &= 0 \\ x_2 + x_3 + x_4 &= 0 \end{aligned}$$

x_3, x_4 frei wählbar. Damit folgt:

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

$\mathcal{C} = \{(y_1, ..., y_4) : y_2 + y_3 = 0, y_1 + y_2 + y_4 = 0\}$

10.13 Satz

Sei $\mathcal{C} \neq \{0\}$ ein [n,k]-Code über K und Kontrollmatrix H. Dann:

$$\begin{aligned} d(\mathcal{C}) = wt(\mathcal{C}) &= min\{r \in \mathbb{N} : \text{ es gibt r linear abhängige Spalten in H}\} \\ &= max\{r \in \mathbb{N} : \text{ je r-1 Spalten von H sind linear unabhängig}\} \end{aligned}$$

10.13.1 Beweis

$s_1, ..., s_n$ Spalten von H. $\mathcal{C} + \{0\} \Rightarrow k \geq 1$.

$rg(H) =$ n-k $<$ n $\Rightarrow s_1, ..., s_n$ sind linear abhängig.

Sei w minimal, so dass es w linear abhängige Spalten gibt: $s_{i_1}, ..., s_{i_w}$
$\exists c_{i_j} \in K : \; c_{i_j} s_{i_1} + ... + c_{i_w} s_{i_w} = 0$

Aus der Minimalität von w folgt: $c_{i_j} \neq 0$ für $j = 1, ..., w$
Setze $c_l = 0$ für $l \in \{1, ..., n\} \setminus \{i_1, ..., i_w\}$.

$c = (c_1, ..., c_n)$

Behauptung:

$$H \cdot c^t = 0 \Leftrightarrow c \cdot H^t = 0$$

$$c \cdot H^t = \begin{pmatrix} s_1 \\ ... \\ s_n \end{pmatrix} = c_1 s_1 + ... + c_n s_n = 0$$

$c \in \mathcal{C}$, $wt(c) = w$.

$wt(\mathcal{C}) \leq w$ Falls $\tilde{c} \in \mathcal{C}$ ex. mit $wt(\tilde{c}) = \tilde{w} < w$, so folgt mit demselben Argument, dass es in H $\tilde{w}$ Spalten gibt, die lin. abhängig sind. Widerspruch!

$wt(\mathcal{C}) = w$.

10.14 Beispiel

$\mathcal{C}$ [7,4]-Hamming-Code über $\mathbb{Z}_2$.

10.12: Kontrollmatrix von $\mathcal{C}$:

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \tag{10.1}$$

je 2 Spalten von H sind lin. unabhängig.

1./2./4. Spalte sind lin. abhängig.

$d(\mathcal{C}) = 3$

10.15 Satz (Singleton-Schranke)

Ist $\mathcal{C}$ eine linearer [n,k]-Code über K, $d(\mathcal{C}) = d$, so ist

$$d \leq n - k + 1 \text{ (d.h. k} \leq \text{n-d+1)} \tag{10.2}$$

10.15.1 Beweis

Nach 3. Gleichheit in 10.13

$$d \leq rg(H) + 1 = n - k + 1$$

[7,4]-Hamming-Code, d=3, n-k+1 = 7-4+1 = 4

10.16 Bemerkung (Nebenklassen von Unterräumen)

$\mathcal{C}$ Unterraum von Vektorraum V.

$$v \in V : v + \mathcal{C} = \{v + c : c \in \mathcal{C}\} \text{ Nebenklasse von } \mathcal{C} \text{ bezüglich v}$$
$$\text{(affiner Unterraum)}$$

[$v \in v + \mathcal{C}$, da $v = v + \sigma \in v + \mathcal{C}$]

(1) $v_1, v_2 \in V$. Dann:

$$v_1 + \mathcal{C} = v_2 + \mathcal{C} \tag{10.3}$$

oder

$$(v_1 + \mathcal{C}) \cap (v_2 + \mathcal{C}) = \emptyset \tag{10.4}$$

(2) $v_1 + \mathcal{C} = v_2 + \mathcal{C} \Leftrightarrow v_1 - v_2 \in \mathcal{C} \Leftrightarrow v_1 \in v_2 + \mathcal{C}$.
(Insb. $v + \mathcal{C} = \mathcal{C}$ $(= \sigma + \mathcal{C}) \Leftrightarrow v \in \mathcal{C}$)

(3) Wähle aus jeder Nebenklasse einen Vertreter v_i:

$$V = \bigcup v_i + \mathcal{C} \tag{10.5}$$

(4) V endl. dim. VR, K endl. Körper, dim(V) = n, $|K| = q$.
$\mathcal{C}$ k-dim. Unterraum von V.
$|\mathcal{C}| = q^k$, $|v| = q^n$
$v \in V : |v + \mathcal{C}| = |\mathcal{C}|$

(5) Bez. wie in (4). Dann folgt aus (4) und (3):

$$\mathcal{C} \text{ hat genau } q^{n-k} \text{ verschiedene Nebenklassen.}$$

10.17 Syndrom-Decodierung linearer Codes

$\mathcal{C}$ lin. [n,k]-Code über K, $|k| = q$, Kontrollmatrix H, (n-k) $\times$ n -Matrix. Ist $y \in K^n$, so heißt $H \cdot y^t \in K^{n-k}$ Syndrom von y

(a) $x \in \mathcal{C} \Leftrightarrow H \cdot x^t = 0 \Leftrightarrow x$ hat Syndrom 0

(b) y_1, y_2 liegen in der gleichen Nebenklassen bzgl. $\mathcal{C} \Leftrightarrow H \cdot y_1^t = H \cdot y_2^t$ (d.h. y_1, y_2 haben gleiches Syndrom)

$[y_1 \in y_2 + \mathcal{C} \Leftrightarrow y_1 - y_2 \in \mathcal{C} \Leftrightarrow H \cdot y_1^t - H \cdot y_2^t = H(y_1 - y_2)^t = 0 \Leftrightarrow H \cdot y_1^t = H \cdot y_2^t]$

(c) Jedes $z \in K^{n-k}$ tritt als Syndrom auf.

$(rg(H) = n - k =$ Rang der lin. Abb. $\begin{cases} y^t \to H \cdot y^t \\ K^n \to K^{n-k} \end{cases}$)

Situtation: $x \in \mathcal{C}$ wird gesendet. $y = x + f$ wird empfangen, f"Fehlervektor".
Nach (b):

$$y \text{ und } f \text{ haben das gleiche Syndrom}$$

Hamming-Decodierung:
y gegeben. Suche $x' \in \mathcal{C}$ mit min. Hamming-Abstand zu y. Das heißt: Suche e mit min. Gewicht, so dass $y - e \in \mathcal{C}$.

e von min. Gewicht in der Nebenklasse $y + \mathcal{C}$, d.h. e von minimalem Gewicht mit $H \cdot y^t = H \cdot e^t$.

Wähle in jeder Nebenklasse ienem Vektor e von minimalem Gewicht: Nebenklassenführer

Liste der Nebenklassenführer.
Ordne Syndrome.
Binär: lexikografisch.
$\begin{pmatrix} 0 \\ ... \\ 0 \end{pmatrix} \leftrightarrow$ Nebenklasse ($\mathcal{C}$) notiere Nebenklasse f.
usw.

y empfangen. Berechne $H \cdot y^t$.

Suche Nebenklassenführer zu diesem Syndrom. Decodiere $y \to y - f \in \mathcal{C}$

Syndrom-Decod. 2^{n-k} Nebenklassenführer $\in \mathbb{Z}_2^n$.
Speicherbedarf: $n \cdot 2^{n-k}$ Bit

Liste aller Codewörter: $n \cdot 2^k$ Bit Speicher

n = 70, k = 50 $\to$ Syndrom-Decod.: $70 \cdot 2^{20}$ Bit $\approx$ 8,75 MB
Liste aller Codewörter: $70 \cdot 2^{50}$ Bit $\approx$ 9 PB

Beispiele guter linearer Codes

11.1 Hamming-Codes

q Primzahlpotenz, K Körper mit $|K| = q$.
Sei $l \in \mathbb{N}, l \geq 2$ mit

$$n = \frac{g^l - 1}{q - 1} \in \mathbb{N}$$

$k = n - l$.
$(q = 2, n = 2^l - 1)$

Dann existiert perfekter [n,k]-Code über K, $d(\mathcal{C}) = 3$, Hamming-Code.

11.1.1 Konstruktion

Es gilt:

$$|K^l \setminus \{\sigma\}| = q^l - 1$$

Deswegen gibt es in K^l genau $\frac{q^l-1}{q-1}$ 1-dim. Unterräume.

Wähle aus jedem 1-dim. Unterraum von K^l einen Vektor $\neq 0$ aus. Schreibe diese als Spaltenvektoren in eine Matrix H. H ist $l \times n$ - Matrix.

Klar: $rg(H) = l$, denn $rg(H) \leq l$, da H nur l Zeilen enthält. $rg(H) \geq l$, da H l linear unabhängige Spalten enthält.

$\mathcal{C}$ = Code mit H als Kontrollmatrix.
$\mathcal{C} = \{y \in K^n : H \cdot y^t = 0\}$ Hamming-Code

$dim(\mathcal{C}) = n - l = k$, $|\mathcal{C}| = q^k$.

Je 2 Spalten von H sind linear unabhängig (denn sonst kämen sie aus demselben 1-dim. Unterraum von K^l). Es gibt 3 linear abhängige Spalten in H:

$$s_1 = \begin{pmatrix} a \\ 0 \\ 0 \\ 0 \end{pmatrix}, s_2 = \begin{pmatrix} 0 \\ b \\ 0 \\ 0 \end{pmatrix}, s_3 = \begin{pmatrix} c \\ c \\ 0 \\ 0 \end{pmatrix}, a, b, c \neq 0, \frac{c}{a} s_1 + \frac{c}{b} s_2 - s_3 = \sigma$$

10.13: $d(\mathcal{C}) = 3$. 1 - Fehler-korrigierender Code (t=1)

Perfekter Code:
$K^n = \bigcup_{c \in \mathcal{C}} K_t(c)$ (disjunkt)

Kugelpackungsschranke mit Gleichheit:

$$|\mathcal{C}| \cdot (1 + n \cdot (q-1)) \stackrel{!}{=} q^n$$

$$q^{n-l} \cdot (1 + q^l - 1) = g^n$$

Damit ist Hamming-Code perfekter Code.

Bei festem q und l gibt es viele Möglichkeiten für H (und damit für $\mathcal{C}$).

- Auswahl der Vektoren $\neq 0$ aus 1-dim. Unterraum von K^l.
- Reihenfolge der Spalten in H.

Führt jeweils zu perfekten Codes mit gleichen Parametern ("äquivalent").

11.2 Beispiel

(a) $q = 2, K = \mathbb{Z}_2 = \{0, 1\}, l = 3, n = 2^3 - 1 = 7, k = n - l = 4.$

[7,4] - Code über $\mathcal{Z}_2$.

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

[7,4] - Hamming-Code zu H über $\mathcal{Z}_2$ = Bsp. aus 9.12.d)

(b) $q = 3, l = 3, n = \frac{3^3-1}{3-1} = 13$

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 2 & 0 & 0 & 1 & 2 & 1 & 1 & 1 & 2 \\ 0 & 1 & 0 & 1 & 1 & 1 & 2 & 0 & 0 & 1 & 1 & 2 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 \end{pmatrix}$$

$dim(\mathcal{C}) = 10, |\mathcal{C}| = 3^{10} = 59.049$

Anwendung: Toto, Tippe alle Codewörter aus $\mathcal{C}$ $\rightarrow$ mind. einmal 12 oder 13 Richtige.

(c) $[7, 4]$ - Hamming-Code über $\mathbb{Z}_2$ aus a).

Ist $y = (1110110) \in \mathcal{C}$?

$$H \cdot y^t = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \text{ d.h. y} \notin \mathcal{C}$$

$\mathcal{C}$ perfekt: Es gibt genau ein Codewort $x \in \mathcal{C}$ mit $d(x, y) = 1$.

Wie findet man x? Es geht schneller als Syndrom-Decodierung:

x unterscheidet sich von y an einer Stelle: $y_i \neq x_i$.

$y = x + (0, ..., 0, \underbrace{1}_{i}, 0, ..., 0)$.

Bilde $H \cdot y^t = Hx^t + H \cdot \begin{pmatrix} 0 \\ ... \\ 1 \\ 0 \\ ... \\ 0 \end{pmatrix}$ = i-te Spalte von H.

Prüfe an welcher Stelle i Syndrom $H \cdot y^t$ als Spalte in Kontrollmatrix auftritt. Ändere y an Stelle i.

In unserem Beispiel $i = 3$. $y \to x = (1100110) \in \mathcal{C}$

11.3 Decodierung von Hamming-Codes

Für $\mathbb{Z}_2$ wie in 11.2.c). Lässt sich auf bel. Körpern verallgemeinern.

11.4 Definition

K (endl.) Körper. $\langle , \rangle$, $K^n \times K^n \to K$

$u = (x_1, ..., x_n), v = (y_1, ..., y_n) : \langle u, v \rangle = \sum_{i=1}^n x_i y_i$

11.5 Bemerkung

(a) $\langle u, v + w \rangle = \langle u, v \rangle + \langle u, w \rangle$

(b) $\langle u, a \cdot v \rangle = a \cdot \langle u, v \rangle$
Analog im 1. Argument.

(c) $\langle u, v \rangle = \langle v, u \rangle$

(d) $\langle 0, v \rangle = \langle v, 0 \rangle = 0$

(e) Ist $u \in K^n$ mit $\langle u, v \rangle = 0$ für alle $v \in K^n$, so ist $u = 0$.

$(a) - (c) : \langle , \rangle$ ist symmetrische Bilinearform.

11.5.1 Beweis

(e):
$0 = \langle u, e_i \rangle = x_i, i = 1, ..., n$ mit $u = (x_1, ..., x_n)$

Beachte: Aus $\langle v, v \rangle = 0$, folgt nicht $v = 0$.

Zum Beispiel: $\mathbb{Z}_2 : \langle v, v \rangle = 0 \Leftrightarrow \sum_{i=1}^n y_i^2 = 0$

$v = (y_1, ..., y_n) : \sum_{i=1}^n y_i = 0 \Leftrightarrow$ Anzahl der Einsen in v ist gerade ($\mathbb{Z}_2$)

11.6 Definition

Sei $\mathcal{C} \subseteq K^n$ ($\mathcal{C}$ braucht kein Unterraum zu sein). Dann heißt

$$\mathcal{C}^\perp = \{y \in K^n : \langle y, x\rangle = 0 \text{ für alle } x \in \mathcal{C}\}$$

der duale (oder orthogonale) Code zu $\mathcal{C}$.
$\mathcal{C}^\perp$ ist Unterraum, auch wenn $\mathcal{C}$ kein Unterraum ist.

11.7 Satz

Sei $\mathcal{C}$ ein linearer [n,k]-Code über K.

(a) $\mathcal{C}^\perp$ ist linearer [n, n-k]-Code

(b) $(\mathcal{C}^\perp)^\perp = \mathcal{C}$

(c) G Erzeugermatrix von $\mathcal{C} \Leftrightarrow G$ ist Kontrollmatrix von $\mathcal{C}^\perp$
H Kontrollmatrix von $\mathcal{C} \Leftrightarrow H$ ist Erzeugermatrix von $\mathcal{C}^\perp$

(d) Hat $\mathcal{C}$ Erzeugermatrix (E_k, A) in Standardform, so ist

$$(-A^t, E_{n-k})$$

eine Erzeugermatrix von $\mathcal{C}^\perp$.

11.7.1 Beweis

(a) G Erzeugermatrix von $\mathcal{C}$, $G = \begin{pmatrix} x_1 \\ ... \\ x_k \end{pmatrix}$, x_i Basis von $\mathcal{C}$. Sei $y \in K^n$.

$$\begin{aligned} y \in \mathcal{C}^\perp &\Leftrightarrow \langle x, y\rangle = 0 \text{ für alle } x \in \mathcal{C} \\ &\Leftrightarrow \langle x_i, y\rangle = 0 \text{ für alle i=1,...,k} \\ &\Leftrightarrow x_i \cdot y^t = 0 \text{ für alle i=1,...,k} \\ &\Leftrightarrow G \cdot y^t = 0 \end{aligned}$$

D.h. G ist Kontrollmatrix von $\mathcal{C}^\perp$.

$$dim(\mathcal{C}^\perp) = n - rg(G) = n - k$$

(b) $\mathcal{C} \subseteq (\mathcal{C}^\perp)^\perp$

$$dim((\mathcal{C}^\perp)^\perp) \underset{(a)}{=} n - dim(\mathcal{C}^\perp) = n - (n-k) = k = dim(\mathcal{C})$$

Also: $\mathcal{C} = (\mathcal{C}^\perp)^\perp$

(c) Aus (a): G Erzeugermatrix von $\mathcal{C} \Rightarrow$ G Kontrollmatrix von $\mathcal{C}^\perp$

Sei G Kontrollmatrix von $\mathcal{C}^\perp$.

$$G = \begin{pmatrix} x_1 \\ \dots \\ x_k \end{pmatrix}$$

Dann: $G \cdot y^t = 0$ für alle $y \in \mathcal{C}^\perp$, d.h. $\langle x_i, y \rangle = 0, i = 1, ..., k$, für alle $y \in \mathcal{C}^\perp$.

Damit:

$$x_1, ..., x_k \in (\mathcal{C}^\perp)^\perp \underset{(b)}{=} \mathcal{C}$$

Zeilen sind lin. unabhängig, $dim(\mathcal{C}) = k \Rightarrow x_1, ..., x_k$ Basis von $\mathcal{C}$.

$\Rightarrow G$ ist Erzeugermatrix von $\mathcal{C}$.

Zweite Äquivalenz folgt aus der ersten und (b).

(d)

$$\begin{aligned}(-A^t, E_{n-k}) \cdot (E_k, A)^t &= (-A^t, E_{n-k}) \cdot \begin{pmatrix} E_k \\ A^t \end{pmatrix} \\ &= -A^t \cdot E_k + E_{n-k} \cdot A^t \\ &= 0\end{aligned}$$

$\Leftrightarrow (-A, E_{n-k}) \cdot x^t = 0$ für alle $x \in \mathcal{C}$.

$(-A^t, E_{n-k})$ ist Kontrollmatrix von $\mathcal{C}$.

11.8 Simplex-Codes

Simplex-Codes sind die dualen Codes der Hamming-Codes.
$|K| = q$, $l \in \mathbb{N}, l \geq 2$.

Hamming-Code $\mathcal{H}$ ist $[\underbrace{\frac{q^l - 1}{q - 1}}_{n}, n - l]$-Code.

Kontrollmatrix H von $\mathcal{H}$ enhlt aus jedem 1-dim. des K^l einen Vektor $\neq 0$ als Spalte.
Sei $\mathcal{C} = \mathcal{H}^\perp$. 11.7.(c) H ist Erzeugermatrix von $\mathcal{C}$.

$\mathcal{C}$ ist $[\frac{q^l-1}{q-1}, l]$-Code Simplex-Code

11.9 Satz

Jedes Codewort $\neq 0$ im Simplex-Code $\mathcal{C}$ hat Gewicht q^{l-1}, d.h. der Abstand zwischen zwei verschiedenen Codewörtern ist (konstant) q^{l-1}.

Insbesondere ist $d(\mathcal{C}) = q^{l-1}$. $\mathcal{C}$ ist $\frac{q^{l-1}-1}{2}$ Fehlerkorrigierend.

Ist $q = 2$: $\mathcal{C}$ ist $(2^{l-2} - 1)$-Fehler korrigierend

11.9.1 Beweis

Seien $z_1, ..., z_l$ Zeilen von H.
$0 \neq x = (x_1, ..., x_n \in \mathcal{C})$.

$$x = \sum_{i=1}^{l} a_i z_i, \ a_i \in K$$

Setze $a = (a_1, ..., a_l) \in K^l$. Es ist $a \neq 0$.

$\langle a \rangle^\perp$ in K^l. $\langle a \rangle^\perp$ hat Dim. $l - 1$ nach 11.7.(a).

$\langle a \rangle^\perp$ enhält $\frac{q^l-1}{q-1}$ 1-dim. Unterräume des K^l.

$\langle a \rangle^\perp$ enhält $\frac{q^l-1}{q-1}$ Spalten h_j von H.

Es gilt:

$$x_j = 0 \Leftrightarrow \sum_{i=1}^{l} a_i z_{ij} = 0 \Leftrightarrow \begin{pmatrix} z_{1j} \\ ... \\ z_{lj} \end{pmatrix} \leftarrow \text{ j-te Spalten in H} \in \langle a \rangle^\perp$$

Damit sind in x genau $\frac{q^{l-1}-1}{q-1}$ Komp. $x_j = 0$. Also

$$wt(x) = n - \frac{q^{l-1} - 1}{q - 1} = \frac{q^l - 1}{q - 1} - \frac{g^{l-1} - 1}{q - 1} = q^{l-1}$$

11.10 Reed-Solomon-Codes

Sei K endl. Körper, $|K| = q$. Wähle k, n mit $1 \leq k \leq n \leq q$.

$$K[x]_k = \{f \in K[x] : \text{ Grad}(f) < k\}$$

ist K-VR der Dimension k (Basis: $1, x, x^2, ..., x^{k-1}$).

Wähle $\mathcal{M} = (a_1, ..., a_n), a_i \in K, a_i \neq a_j$ für $i \neq j$ (geht, da $n \leq q$)

Setze $\mathcal{C}_\mathcal{M} = \{f(a_1), ..., f(a_n)) : f \in K[x]_k\} \subseteq K^n$ (allgemeiner) Reed-Solomon-Code

$\mathcal{C}_\mathcal{M}$ ist sog. Auswertungscode.

11.11 Satz

(a) $\mathcal{C}_{\mathcal{M}}$ ist linearer [n,k]-Code.

(b) $d(\mathcal{C}_{\mathcal{M}}) = n - k + 1$

(c) $G = \begin{pmatrix} 1 & ... & 1 \\ a_1 & ... & a_n \\ a_1^2 & ... & a_n^2 \\ ... & ... & ... \\ a_1^{k-1} & ... & a_n^{k-1} \end{pmatrix}$ ist Erzeugermatrix von $\mathcal{C}_{\mathcal{M}}$

11.11.1 Beweis

(a) $\mathcal{C}_{\mathcal{M}}$ ist linear ✓

$$\begin{aligned}(f(a_1), ..., f(a_n)) + (g(a_1), ..., g(a_n)) &= (f(a_1) + g(a_1), ..., f(a_n) + g(a_n)) \\ &= ((f+g)(a_1), ..., (f+g)(a_n)) \in \mathcal{C}_{\mathcal{M}}, \text{ da Grad(f+g)} < \text{k}\end{aligned}$$

Analog bzgl. Mult. mit Skalaren.

Abb.

$$\alpha : \begin{cases} K[x]_k \to \mathcal{C}_{\mathcal{M}} \\ f \mapsto (f(a_1), ..., f(a_n)) \end{cases}$$

ist lineare Abb., surjektiv.

α injektiv: Sei $f \in K[x]_k, f \in ker(\alpha)$, d.h. $(f(a_1), ..., f(a_n)) = (0, ..., 0)$

Also: f hat n versch. Nullstellen $n \geq k > k - 1 \geq \text{Grad}(f) \to f = 0$.
α bijektive lin. Abb. $\Rightarrow dim(\mathcal{C}_{\mathcal{M}}) = dim(K[x]_k) = k$.

$\mathcal{C}_{\mathcal{M}}$ ist [n,k]-Code.

(b) Jedes $f \in K[x]_K$, $f \neq 0$, hat höchstens $k - 1$ Nullstellen in K.
In $c = (f(a_1), ..., f(a_n))$ sind höchstens $k - 1$ Stellen gleich 0. D.h. $wt(c) \geq n - (k - 1) = n - k + 1$ für alle $0 \neq c \in \mathcal{C}_M$

$$d(\mathcal{C}_M) = wt(\mathcal{C}_M) \geq n - k + 1$$

Setze $f = (x - a_1) \cdot ... \cdot (x - a_{k-1}) \in K[x]_k$. $c = (f(a_i), ..., f(a_n)) \in \mathcal{C}_M$ hat Nullen an den ersten $k - 1$ Stellen (und sonst keine).

$$wt(c) = n - k + 1$$

Damit folgt:

$$d(\mathcal{C}_M) = wt(\mathcal{C}_M) = n - k + 1$$

(Folgt auch aus Singleton-Schranke)

(c) $1, x, ..., x^{k-1}$ bilden Basis von $K[x]_k$.

$\underset{(a)}{\Rightarrow} \underbrace{\alpha(1), \alpha(x), ..., \alpha(x^{k-1})}_{\text{Zeilen von G}}$ Basis von $\mathcal{C}_m$

11.12 Bemerkung

(a) Für $q = 2$ ist $n \leq 2$, d.h. die Codes sind trivial.

(b) Für jeden Code gilt Singleton-Schranke: $d \leq n - k + 1$.
Für Reed-Solomon-Codes gilt Gleichheit:

$$MDS - Codes$$

(maximum-distance-separable)

11.13 Die speziellen Reed-Solomon-Codes

$n = q - 1$. Sei α ein Erzeugendes der mult. Gruppe $K^* = K \setminus \{0\} = \{\alpha^0 = 1, \alpha^1 = \alpha, \alpha^2, ..., \alpha^{q-2}\}$ $(\alpha^{q-1} = 1)$

Setze $a_i = \alpha^{i-1}$ $(\mathcal{M} = (\alpha^0 = 1, \alpha, ..., \alpha^{q-2}))$
Der entsprechende Reed-Solomon-Codes heißt

$$RS_q(d)$$

Mann gibt sich d vor. $n = q - 1$. $k = n - d + 1$
Nach 11.11.c) ist

$$G = \begin{pmatrix} 1 & 1 & 1 & ... & 1 \\ 1 & \alpha & \alpha^2 & ... & \alpha^{q-2} \\ 1 & \alpha^2 & \alpha^4 & ... & \alpha^{2\cdot(q-2)} \\ ... & ... & ... & .... & ... \\ 1 & \alpha^{k-1} & \alpha^{2\cdot(k-1)} & ... & \alpha^{(q-2)\cdot(k-1)} \end{pmatrix}$$

Erzeugermatrix von $RS_q(d)$.

11.13.1 Beispiel

$q = 7.$, $n = 6$, $d = 4$, $k = 3$.
$K = \mathbb{Z}_7 = \{0, 1, 3, 3^2 = 2, 3^3 = 6, 3^4 = 4, 3^5 = 5\}$ $(\alpha = 3)$

$RS_7(4)$:

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 2 & 6 & 4 & 5 \\ 1 & 2 & 4 & 1 & 2 & 4 \end{pmatrix}$$

$$\underbrace{\alpha^i \cdot \underbrace{(1, \alpha^i, \alpha^{2i}, ..., \alpha^{i\cdot(q-2)})}_{\in\mathcal{C}}}_{\in\mathcal{C}} = (\alpha^i, \alpha^{2i}, ..., \alpha^{i\cdot(q-2)}, \underbrace{1}_{\alpha^i\cdot\alpha^{i\cdot(q-2)}}) \in \mathcal{C}$$

Zyklischer Shift um 1 Stelle (nach links) eines Codeworts ergibt wieder ein Codewort. Folge: Zyklischer Shit um bel. viele Stellen ergibt wieder ein Codewort.

$RS_q(d)$ ist ein sog. zyklischer Code

11.13.2 Beispiel

$q = 2^8 = 256$, $n = 255$, $k = 125$, $d = 131$, 65-Fehler-korrigierend.

$z \in (K_q)^{255} \underset{\text{decod.}}{\rightarrow} \text{c}$

$|\mathcal{C}| = q^k = 256^{125} = 2^{1000}$

Syndrom Decod.: 256^{130}

Fazit: Syndrom-Decodierung oder "Liste durchgehen" bieten sich nicht an.

Interleaving und Audio-CD-Codierung

CD's: Typische Fehler Auslöschungen als "Bursts"

12.1 Lemma

Ein linearer Code mit Minimalabstand d kann bis zu $d-1$ Auslöschungen korrigieren, wenn keine weiteren Fehler aufgetreten sind.

12.1.1 Beweis

$c = (c_1, ..., c_n)$ gesendet. $z = (c_1, ..., *, ..., *, ..., c_n)$ empfangen. $*$ = Auslöschungen an $a \leq d-1$ Stellen.

ist c' Codewort, das an den nicht ausgelöschten Stellen mit z übereinstimmt, so gilt:

$$d(c, c') \leq a \leq d-1 \Rightarrow c = c'$$

12.2 Cross-Interleaving

Technik zur Korrektur von "Burst"-Fehlern.

(a) Geg.: $\mathcal{C}_1, \mathcal{C}_2$ $[n_i, k_i]$-Codes, $i = 1,2$ über K mit Erzeugermatrizen in Standardform.

$$(E_{k_i} | \underbrace{A_i}_{n_i - k_i})$$

Gegeben seien k_1 Infowörter der Länge k_2.

$$i_1 = (c_{11}, ..., c_{k_2 1})$$

$$...$$

$$i_{k_1} = (c_{1k_1}, ..., c_{k_2 k_2})$$

Codiere $i_1, ..., i_{k_1}$ mit dem <u>äußeren Code</u> $\mathcal{C}$ durch Multiplikation von i_j mit $(E_{k_2} | A_2)$. Liefert:

$$c_1 = (\underbrace{i_1}_{k_2}, \underbrace{*...*}_{n_2 - k_2}), ..., c_{k_1} = (i_{k_1}, *...*) \in \mathcal{C}_2$$

Schreibe $c_1, ..., c_{k-1}$ als Spalten in eine Matrix

$$\begin{pmatrix} i_1^t & i_2^t & ... & i_{k_1}^t \\ * & * & ... & * \\ ... & ... & ... & ... \\ * & * & * & * \end{pmatrix} n_2 \times k_1 - Matrix$$

Codiere die Zeilen mit dem <u>inneren Code</u> $\mathcal{C}_1$ durch Multiplikation mit $(E_{k_1}|A_1)$.

$$\begin{pmatrix} i_1^t & ... & i_{k_1}^t & * \\ * & ... & * & * \end{pmatrix} n_2 \times n_1 - Matrix$$

Letzen $n_1 - k_1$ Spalten $\in \mathcal{C}_\in$, da sie Lin. Komb. der ersten k_1 Spalten sind.

Lese Matrix zur Übertragung / Speicherung zeilenweise aus: Wort der Länge $n_1 \cdot n_2$.

$$w = |\overbrace{\underbrace{-------|}_{\in \mathcal{C}_1}\underbrace{--------|}_{\in \mathcal{C}_1}....}^{i_1}|-------|...|\underbrace{-------|}_{\in \mathcal{C}_1}$$