# Zusammenfassung M114

Codierungs-, Kompressions- und Verschlüsselungsverfahren einsetzen

2008-11-12

Emanuel Duss

# Über

| | |
|---|---|
| Autor | Emanuel Duss |
| Erstellt | 2008-07-24 |
| Bearbeitet | 2008-11-12 |
| Heute | 2008-11-12 |
| Bearbeitungszeit | 14:55:23 |
| Lehrjahr des Moduls | 1. Lehjahr 2006/2007 |
| Pfad | / home/emanuel/Daten/Lehre/Zwischenprüfungen/Zusammenfassungen_von_mir/ M114/M114_Zusammenfassung.odt |
| CC-Lizenz |  Creative Commons Namensnennung-Keine kommerzielle Nutzung-Weitergabe unter gleichen Bedingungen 2.5 Schweiz http://creativecommons.org/licenses/by-nc-sa/2.5/ch/ |
| Powered by |  |

# Bearbeitungsprotokoll

| Datum | Änderung(en) |
|---|---|
| 2008-07-24 | Erstellt |
| 2008-08-02 | Überarbeitet, Stichwortverzeichnis erstellt |
| 2008-08-19 | Fertigstellung der Zusammenfassung |

# Inhaltsverzeichnis

# Modulbaukasten



## Modulidentifikation

| | |
|---|---|
| Modulnummer | 114 |
| Titel | Codierungs-, Kompressions- und Verschlüsselungsverfahren einsetzen |
| | |
| Kompetenz | Codierungs-, Kompressions- und Verschlüsselungsverfahren einsetzen |
| | |
| Handlungsziele | 1. Geeignete Kompressionsverfahren für die Archivierung von Information auswählen und einsetzen. Dabei die erforderlichen Vorkehrungen für eine langfristige Wiederverwendung der Informationen beachten und die dazu erforderlichen Massnahmen treffen. |
| | 2. Codierungen und Kompressionsverfahren für die Übertragung von Informationen auswählen und Voraussetzungen definieren, die einen problemlosen Austausch codierter und komprimierter Informationen ermöglichen. |
| | 3. Verschlüsselungsverfahren zur Sicherung von Informationen gegen unbefugten Zugriff auf Speichern und Übertragungswegen auswählen und einsetzen. |
| | 4. Gesicherte Übertragungsverfahren für Dateien mit asymmetrischen und symmetrischen Verschlüsselungsverfahren nutzen. Dabei Aspekte wie Public/Private Key, Zertifikate, Protokolle und Standards berücksichtigen. |
| | |
| Kompetenzfeld | IT Sicherheit |
| Objekt | Zu speichernde und zu übertragende Informationen. |
| Niveau | 2 |
| Voraussetzungen | keine |
| Anzahl Lektionen | 40 |
| Anerkennung | Eidg. Fähigkeitszeugnis Informatiker/Informatikerin |
| | |
| Modulversion | 1.2 |
| MBK Release | R3 |
| Harmonisiert am | 29.09.2005 |

## Handlungsnotwendige Kenntnisse

Handlungsnotwendige Kenntnisse beschreiben Wissenselemente, die das Erreichen einzelner Handlungsziele eines Moduls unterstützen. Die Beschreibung dient zur Orientierung und hat empfehlenden Charakter. Die Konkretisierung der Lernziele und des Lernwegs für den Kompetenzerwerb sind Sache der Bildungsanbieter.

| | |
|---|---|
| Modulnummer | 114 |
| Titel | Codierungs-, Kompressions- und Verschlüsselungsverfahren einsetzen |
| Kompetenzfeld | IT Sicherheit |
| Modulversion | 1.2 |
| MBK Release | R3 |

| Handlungsziel | Handlungsnotwendige Kenntnisse |
|---|---|
| 1. | 1. Kennt die wichtigsten Typen von Binärcodes (ANSI-, BCD-, EAN-,1-aus -n-, Gray- und Huffman-Code usw.) und kann anhand ihrer Merkmale (Zeichenvorrat, Redundanz) aufzeigen, wie sich diese hinsichtlich der Bewertbarkeit, Fehlererkennbarkeit und Rechenfähigkeit unterscheiden.<br>2. Kennt die wichtigsten Eigenschaften von Bildern (S/W-Strichzeichnung, Farbfoto, bewegte/nicht bewegte Bilder, vektorisiert/pixelorientiert usw.) und kann erläutern, wie damit die Bildqualität (Auflösung, Farbtiefe), der Bildaufbau und der Speicherbedarf beeinflusst werden kann.<br>3. Kennt das Prinzip der Kompression von Daten und kann an Beispielen aufzeigen, welche Wirkung damit bei der Speicherung von Daten erzielt werden kann.<br>4. Kann für die gängigen Inhalte von Dokumenten (Text und Grafik) aufzeigen, wie deren Anteil den Kompressionsgrad mit einem Kompressionsverfahren (PKZIP, gzip, stuffit) beeinflussen. |
| 2. | 1. Kennt die gängigen Kompressionsverfahren für die Übertragung von Daten und kann anhand ausgewählter Normen/Standards (JPG, PNG, MPG, H261/263, ZIP usw.) erläutern, in welchen Anwendungsbereichen diese eingesetzt werden. |
| 3. | 1. Kennt das grundsätzliche Prinzip der Verschlüsselung von Informationen (Kryptografie/Steganografie) und kann anhand eines einfachen Verschlüsselungskonzepts aufzeigen, wie damit Informationen chiffriert und dechiffriert werden können. |
| 4. | 1. Kennt die prinzipiellen Unterschiede zwischen einer symmetrischen und asymmetrischen Verschlüsselung (Passwörter, private und öffentliche Schlüssel) und kann erläutern, wie sich diese auf den Grad der Datensicherheit auswirken.<br>2. Kennt das Prinzip elektronischer Signatursysteme und kann anhand von Beispielen aufzeigen, wie damit die Sicherheit (Authentifizierung, Autorisierung) der Übermittlung gewährleistet werden kann.<br>3. Kennt den Zweck digitaler Zertifikate und kann an Beispielen erläutern, wie damit das Vertrauen zwischen Anbieter und Bezüger (einer Leistung, eines Produkt) sichergestellt werden kann. |

Der Block 3 „Grundlagen Schaltnetze“ wird an der Zwischenprüfung nicht mehr geprüft!

# 1 Zahlensysteme

## 1.1 Zahlensysteme in der Informatik

- In der Informatik muss man Informationen darstellen können
- Die einfache physikalische Realisierung erfolgt durch 0 und 1 in der Informatik (Spannung oder keine Spannung).

## 1.2 Geschichte

- 30'000 v. Chr. Begann man zu Zählen (mit Kerbholz, Knochen mit Kerben, Steinen, Körner, Stäbchen.
- Menschen wollten Zahlen aufschreiben. Für jede Zahl ein Zeichen gäbe zu viele Zeichen :-)
- Unsere Vorfahren machten für die zu zählende Anzahl so viele Striche. Das war aufwändig und unübersichtlich
- Die Ägypter machten 10er Bündel. 10 Einheiten hatten ein anderes Zeichen. So auch bei 100 und 1000, etc...

### 1.2.1 Römisches Zahlensystem

#### Grundsymbole

| | |
|---|---|
| I | 1 |
| V | 5 |
| X | 10 |
| L | 50 |
| C | 100 |
| D | 500 |
| M | 1000 |

#### Regeln

- Links ist die grösste Zahl
- I, X, C werden bis drei mal geschrieben
- VLD werden nur einmal geschrieben
- Steht ein Symbol einer kleineren Zahl vor dem einer grösseren, so wird sein Wert von dem folgenden subtrahiert.

#### Nachteile

- Schwierig zu Rechnen
- Unübersichtlich bei grossen Zahlen
- Keine Null.

# 1.3 Zahlensysteme mit Stellenschreibweise

- Deutliche Verbesserung
- Die Stellung eines Symbols bestimmt den Wert:

| 3 Wochen | 4 Tage | 2 Stunden | 0 Minuten | 23 Sekunden | Total |
|---|---|---|---|---|---|
| 3 * (7*24*60*60) | 4 * (24*60*60) | 2 * (60*60) | 0 * (60) | 23 * (1) | 2167223 Sekunden |

# 1.4 Polyadisches Zahlensystem

- Wie Stellenschreibweise, zusätzlich werden die einzelnen Stellenwerte nach einen exponentiellen Bildungsgesetz eingestuft.

| | |
|---|---|
| **Basis** | Wie viele Zeichen sind vorhanden? (Dualsystem hat die Basis 2) |
| **Ziffer** | Ein Zeichen aus dem Zeichenvorrat |
| **Zeichenvorrat** | Welche Zeichen es alles kein können (Dualsystem: Zeichenvorrat = {0, 1}) |
| **Stellenwert** | Wie hoch die Stelle ist. Bei **5**23 wäre es $10^2 = 100$ |
| **Nennwert** | Zahl aus dem Zeichenvorrat: Bei **5**23 wäre es 5. |
| **Ziffernwert** | Produkt von Nennwert uns Stellenwert. Bei **5**23 wäre es 5 * 100 = 500. |
| **Zahl** | Folge von Ziffern. |
| **Wert** | Summe der einzelnen Ziffernwerte |

## 1.4.1 Beispiele aus der Praxis

| **Zahlensystem** | **Dezimales** | **Binäres** | **Oktales** | **Hexadezimales** |
|---|---|---|---|---|
| **Basis** | Basis = 10 | Basis = 2 | Basis = 8 | Basis = 16 |
| **Ziffern** | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 | 0, 1 | 0, 1, 2, 3, 4, 5 6, 7 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F |
| **Info** | Von Adam Riese | An, Aus. Einfache Schaltungen in Rechnern Braucht viele Stellen. | Einfache Umrechnung von Binär in Oktal. | Einfache Umrechnung von Binär in Hexadezimal |

> Bitte immer an die Syntax denken: Jede Zahl hat am Schluss die Basis als tief gestellt Zahl. (Ausser beim Dezimalsystem)

# 1.5 Zahlensysteme umrechnen

Immer die Basis angeben!!

## 1.5.1 Dezimalsystem in beliebiges

207,22 in das Dualsystem

| **Ganzzahliger Bereich** | **Nachkommateil** |
|---|---|
| 207 : 2 = 103 Rest: 1<br>103 : 2 = 51 Rest: 1<br>51 : 2 = 25 Rest: 1<br>25 : 2 = 12 Rest: 1<br>12 : 2 = 6 Rest: 0<br>6 : 2 = 3 Rest: 0<br>3 : 2 = 1 Rest: 1<br>1 : 2 = 0 Rest: 1 | 2 · 0,22 = 0,44 --> Ziffer: 0<br>2 · 0,44 = 0,88 --> Ziffer: 0<br>2 · 0,88 = 1,76 --> Ziffer: 1<br>2 · 0,76 = 1,52 --> Ziffer: 1<br>2 · 0,52 = 1,04 --> Ziffer: 1<br>2 · 0,04 = 0,08 --> Ziffer: 0<br>2 · 0,08 = 0,16 --> Ziffer: 0<br>2 · 0,16 = 0,32 --> Ziffer: 0 |
| **Resultat (von unten nach oben): 11001111** | **Resultat (von oben nach unten): 00111000** |
| **Zusammenfassen: 11001111,00111000$_2$** | |

## 1.5.2 Beliebiges ins Dezimalsystem

| **Ganzzahliger Bereich (11001111)** | **Nachkommateil (0.0011110)** |
|---|---|
| Man muss die Potenz beachten. Beim Dualsystem (Bais 2) ist es $2^1$, $2^2$, $2^3$, $2^4$, $2^5$, $2^6$ etc...<br>1 · 1 = 1<br>1 · 2 = 2<br>1 · 4 = 4<br>1 · 8 = 8<br>0 · 16 = 0<br>0 · 32 = 0<br>1 · 64 = 64<br>1 · 128 = 128<br>207 | Man wandelt die Zahl 0011110 ins Dezimalsystem um:<br>0 · 1 = 0<br>1 · 2 = 2<br>1 · 4 = 4<br>1 · 8 = 8<br>1 · 16 = 16<br>0 · 32 = 0<br>0 · 64 = 0<br>30<br>Und teilt sie durch die nächst höhere Potenz (hier im Beispiel ist es 128).<br>30 : 128 = 0,234375 |
| **Zusammenfassen: 207,234375** | |

## 1.5.3 Dualsystem zu Octalsystem und umgekehrt

$00110101100101001 0_2$; Man bildet 3-er Gruppen, und wandelt diese gemäss Tabelle um!

| Binärsystem | 001 | 101 | 011 | 001 | 010 | 010 |
|---|---|---|---|---|---|---|
| Oktalsystem | 1 | 5 | 3 | 1 | 2 | 2 |

### 1.5.4 Dualsystem zu Hexadezimalsystem und umgekehrt

$00110101100101001 0_2$; Man bildet 4er-Gruppen, und wandelt diese gemäss Tabelle um.

| Dualsystem | 00 | 1101 | 0110 | 0101 | 0010 |
|---|---|---|---|---|---|
| Hexadezimalsystem | 0 | 13 = D | 6 | 5 | 2 |

### 1.5.5 Eine kleine super Umrechnungstabelle

| Dual | Octal | Dez | Hex |
|---|---|---|---|
| **0** | **0** | **0** | **0** |
| **1** | **1** | **1** | **1** |
| 10 | **2** | **2** | **2** |
| 11 | **3** | **3** | **3** |
| 100 | **4** | **4** | **4** |
| 101 | **5** | **5** | **5** |
| 110 | **6** | **6** | **6** |
| 111 | **7** | **7** | **7** |
| 1000 | 10 | **8** | **8** |
| 1001 | 11 | **9** | **9** |
| 1010 | 12 | 10 | **A** |
| 1011 | 13 | 11 | **B** |
| 1100 | 14 | 12 | **C** |
| 1101 | 15 | 13 | **D** |
| 1110 | 16 | 14 | **E** |
| 1111 | 17 | 15 | **F** |
| 10000 | 20 | 16 | 10 |

### 1.5.6 Zahlensysteme umrechnet mit dem Ti Voyage 200

| | | |
|---|---|---|
| Von Dezimal: kein Präfix: 523<br>Von Binär: **0b**001011<br>von Hexadezimal: **0h**0f89d | $2^{nd}$ + ▶<br>(Das ▶ erreicht man durch 2nd + Y.) | dec<br>hex<br>bin |

Immer die Basis angeben!!

# 2 Rechnen mit Dualzahlen

## 2.1 Rechenoperationen

### 2.1.1 Addition

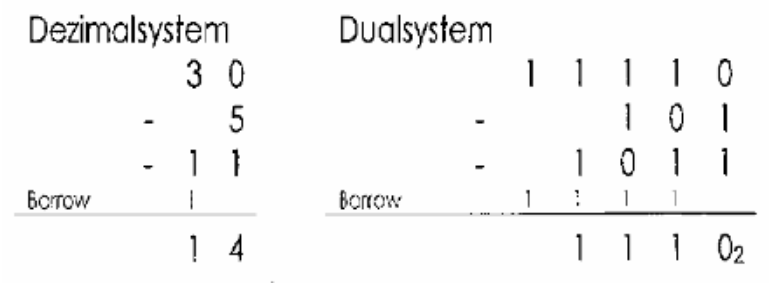
- 0 + 0 = 0; 0 + 1 = 1; 1 + 0 = 1; 1 + 1 = 0
- Achtung: 1 + 1 = 0 --> das Übertragungsbit erhält den Status 1

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | 1 | 0 | 1 | 0 | 0 | 1 |
| | | + | 1 | 1 | 0 | 1 | 1 | 0 |
| | | + | 0 | 1 | 0 | 1 | 0 | 1 |
| | | + | 0 | 1 | 0 | 0 | 1 | 0 |
| | | + | 1 | 0 | 1 | 0 | 1 | 0 |
| | | + | 0 | 1 | 0 | 0 | 1 | 0 |
| | | + | 1 | 0 | 0 | 1 | 1 | 0 |
| *Carry-Bit* | *1* | *3* | *3* | *2* | *3* | *3* | *1* | |
| Resultat | 1 | 1 | 1 | | 1 | | 0 | 0 |
| Kommentar | 1 | 3 / 2 = 1 Rest 1 | 7 / 3 = 3 Rest 1 | 6 / 2 = 3 Rest 0 | 5 / 2 = 2 Rest 1 | 6 / 2 = 3 Rest 0 | 6 / 2 = 3 Rest 0 | 1 + 1 = 0 Carry 1 |

- Bei gebrochenen Zahlen geht es genau gleich. Das Komma muss jedoch immer am selben Ort bleiben (wie bei dem Dezimalsystem)

### 2.1.2 Subtraktion

0 – 0 = 0
0 – 1 = 1 (1|0)
1 – 0 = 1
1 – 1 = 0




Hier wird so gerechnet bei der ersten Stelle ist eine Null also somit muss eine Zahl von vorne geholt werden also eine 2. 2-2=0 und 1 behalten weil wir es geborgt haben. Bei der Zweiten Zahlenreihe wird wieder eine 2 geholt. Also 3-2 = 1 und behalte 1.

> Die Subtraktion kann mit dem Zweierkomplement durchgeführt werden! Dabei muss addiert werden.

# 2.2 Darstellung von negativen Dualzahlen

## 2.2.1 Übersicht

- Im Binärsystem gibt es keine Vorzeichen!
- Folgendermassen könnte man negative Zahlen darstellen:

### Betrag und Vorzeichen

- Das erste Bit entspricht dem Vorzeichen (0 = positiv; 1 = netativ)
- Diese Darstellung wird in der Digitaltechnik praktisch nie verwendet

### Einer-Komplement

- Sämtliche Bits invertieren
- z.B. +4 = 0100; -4 = 1011
- Ist das erste Bit 1 ist es eine negative Zahl. Es müssen sämtliche Bits erneut invertiert werden.
- Wir sehr selten verwendet!

### Zweier-Komplement / echtes Komplement

- Am häufigsten Verwendet. Alle Mikroprozessoren arbeiten in dieser Darstellung.
- Man kann die Subtraktion mit Hilfe der Addition durchführen: 100 – 55 = 100 + (-55) = 225
- Kann mit jeder polyadischen Zahl gemacht werden
- Beispiel: 3 Stellen (000 bis 999), Komplement der Zahl -55: 10≥ – 55 = 945.
- Test: 310 + 945 = 1255. Da es nur 3 Stellen hat, bekommen wir 255.

> Ist eine Zahl im Zweierkomplement dargestellt, sagt die erste Ziffer aus, ob es sich um eine positive oder negative Zahl handelt.
>
> Das Zweierkomplement einer negativen Dualzahl kann gebildet werden, indem man die Zahl invertiert und 1 addiert

# 3 Digitalisierung

## 3.1 Digitalisierung von Musik und Bild

| | |
|---|---|
| Analog | Kontinuierlich, stufenlos, alle Zwischenwerte werden dargestellt. |
| Samplingrate | Anz. Messungen pro Sekunde (CD: alle 22 Mikrosekunden, also 44'100 mal Pro Sekunde).<br><br>Das menschliche Ohr merkt bei sehr vielen Messungen nicht, dass die Tonspur nur aus vielen Punkten besteht. |
| Quantisierung | Analoges Signal in ein vorgegebenes Raster übertragen (Digitalisierung). |
| Samplingtiefe | Wie viel Bit der Quantisierung zur Verfügung stehen. 16 Bit = $2^{16}$ = 65536 Stufen |
| Abspielen einer CD | Pro Sekunde spielt der CD-Player 88'200 (da Stereo) 16-Stellige Binärzahlen ab. Ausgeschreiben in 0 und 1 wären hätten in der Bibel nur 3.6 Sekunden Musik Platz. |
| Fotos und Videos | Eine gute Kamera speichert pro Bild die Farbwerte von 10'000'000 Bildpunkten (10 Megapixel).<br><br>Auch hier gibt es eine Samplingtiefe: 8 Bit = $2^8$ = 256 Stufen. |

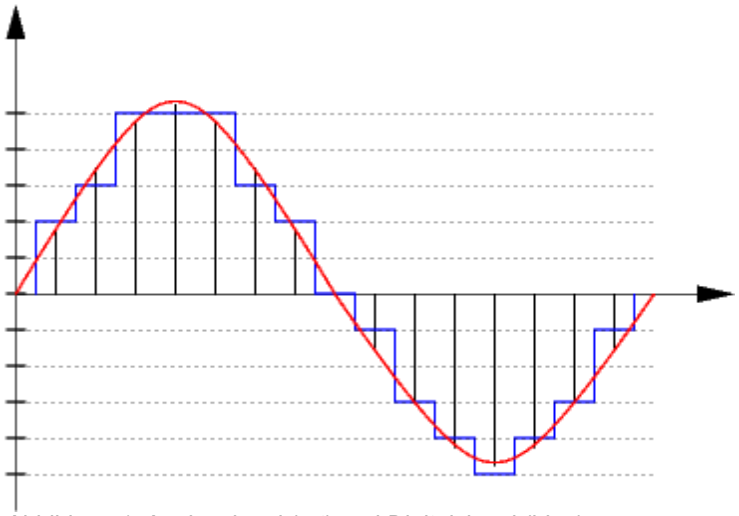
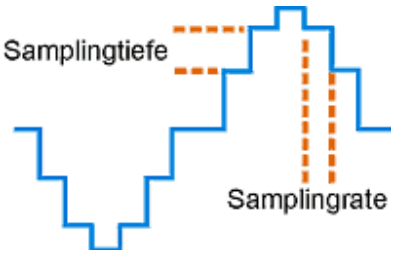


*Abbildung 1: Analogsignal (rot) und Digitalsignal (blau)*



# 3.2 Umwandlung von analogen zu digitalen Signalen

## 3.2.1 Analog-Digital-Wandler

- Analog-Digital-Wandler auch Analog-Digital-Umsetzer oder Analog-Digital-Converter
- Elektronisches Gerät, welches analoge Signale in digitale Daten umwandelt.
- In gleichen Zeitabständen wird beim Eingang die Spannung gemessen (Sampling)
- Digitale Daten werden meist im dualen Zahlensystem oder im BCD-Code ausgegeben.
- Je mehr Bit zur Verfügung stehen, desto grösser ist das Auflösungsvermögen. Das Auflösungsvermögen ist nicht mit der Genauigkeit zu verwechseln.
- Jeder Analog-Digital-Wandler arbeitet mit einer bestimmten Genauigkeit. Es gibt einen Abweichungswert zum Analogsignal (z.B. 10-5).
- Die Abtasthäufigkeit muss mindestens doppelt so gross sein, wie die höchste zuwandelnde Frequenz, die im Analogsignal enthalten ist (z.B. Telefonfrequenz von 20'000 kHz benötigt also 40'000 Abtastvorgänge pro Sekunde; = 40 kHz.)

## 3.2.2 Digital-Analog-Wandler

- Der Digital-Analog-Wandler liest digitale Werte (Binär oder ein Code) aus und zeichnet diese auf. z.B. Aus einer Wertetabelle einen Funktionsgraphen zeichnen.
- Unbewertete Codes müssen zuerst in einen bewerteten Code umgewandelt werden.
- Es können nur Signale eines bestimmten Codes in analoge Signale umgewandelt werden. Der Digital-Analog-Wandler muss für den bestimmten Code geeignet sein.

# 3.3 Speicherplatz

Speicherbedarf = Samplingrate * Samplingtiefe

Der Speicherplatz lässt sich durch Multiplikation von Samplingrate und Samplingtiefe berechnen: z.B.

Samplingrate: 44'100 kHz

Samplingtiefe: 16 Bit --> 216 --> 65'536

Stereo: Faktor 2

Zeit: 60 s

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 44'100 | * | 16 | * | 60 | * | 2 | = | 84672000 | / | 8 | = | 10269 | etwa | 10 |
| 1/s | | Bit | | Pro Minute | | Bei Stereo | | Bit pro Minute | | Byte | | KB pro Min. | | MB pro Minute |

# 4 Codesysteme

- Ein Code ist eine Vorschrift für die eindeutige Zuordnung der Zeichen eines Zeichenvorrats zu denjenigen eines anderen Vorrats
- Ein Beispiel dazu wäre der Morsecode.

## 4.1 EDV-Codesysteme

- In der Informatik werden Informationen fast ausschliesslich binär (0 oder 1) dargestellt.
  - Das ist der einfachen Realisierung von Spannung und keiner Spannung gutzuschreiben.
- Es gibt numerische und alphanumerische Codes.
  - Numerische Codes stellen ausschliesslich Ziffern und Zahlen dar.
  - Alphanumerische Codes stellen auch Buchstaben und Sonderzeichen dar.

### 4.1.1 Anzahl nötige Stellen

Wir haben im Quellsystem 26 verschiedene Zeichen und im Zielsystem zwei verschiedene Zeichen. Wie viele Stellen benötigen wir im Zielsystem?

Anzahl nötige Stellen = Log(Anz. Zeichen Quellsystem) / Log(Anz. Zeichen Zielsystem)

= log(26)/log(2) = 4.7 --> Wir benötigen 5 Stellen (eigentlich 4.7 mit 0.3 Redundanz)

### 4.1.2 Eigenschaften von Codesystemen

| | |
|---|---|
| Stellenzahl | Aus wie vielen Stellen besteht ein Codewort? |
| Bewertbarkeit | Ist jeder Stellen eine bestimmte Wertigkeit zugewiesen? z.B. $2^2$. $2\geq$ etc... |
| Gewicht | Anzahl mit 1 belegten Stellen. z.B. 01011 = Gewicht 2 |
| Minimaldistanz | Minimale Anzahl Stellen, die sich ändern. |
| Maximaldistanz | Maximale Anzahl Stellen, die sich ändern. |
| Hammingdistanz | Anzahl Stellen, die sich ändern zwischen benachbarten Codewörter |
| Stetig | Ist die Distanz zwischen allen Codewörtern konstant? |
| Redundanz | Sind mehr Kombinationen vorhanden als die Codierung mathematisch benötigt |

# 4.2 Nummerische Codes

## 4.2.1 Wortcodes

- Bei Wortcodes wird die Zahl als ganzes codiert.
- Schwieriger zu realisieren, da man nicht Ziffernweise umcodieren kann.
- Mathematische Operationen sind einfacher realisierbar.

### Dualcode

Der Dualcode entspricht der Zahlendarstellung im Dualsystem.

| | |
|---|---|
| Stellenzahl | Unendlich |
| Bewertbarkeit | Ja |
| Gewicht | 0...unendlich |
| Minimaldistanz | 1 |
| Maximaldistanz | unendlich |
| Hammingdistanz | 1...unendlich |
| Stetig | Nein |
| Redundanz | 0 |

## 4.2.2 Zifferncodes

- Bei Zifferncodes werden die Ziffern der Zahl getrennt Codiert
- Einfacher realisierbar, da Ziffernweise umcodiert werden kann.
- Schwieriger für mathematische Operationen
- BCD-Codes: Binary Coded Decimals (Binär kodierte Dezimalziffern). Ziffernorganisierte binäre Codierung. Jeder Dezimalziffer wird ein binäres Codewort zugewiesen. Es werden 4 Bit benötigt.
- Tetrade: Codewort für eine Ziffer

## 8-4-2-1-Code

- Stellt die Ziffern 0-9 im Dualcode dar.
- Mit ihm sind einfache Zähl- und Addierschaltungen möglich.
- Beim Rechnen ist aber eine Korrektur im Falle eines Übertrags (>9) nötig.

| | |
|---|---|
| Stellenzahl | 4 |
| Bewertbarkeit | Ja |
| Gewicht | 0 ... 4 |
| Minimaldistanz | 1 |
| Maximaldistanz | 4 |
| Hammingdistanz | 1 ... 4 |
| Stetig | Nein |
| Redundanz | 0,678 |

## BCD-Zählcode

- Wurde zum Teil in der Fernmeldetechnik beim Wählvorgang angewendet (Impulswahl)
- 1 = 0000000001; 2 = 0000000011; 3 = 0000000111; ...; 9 = 0111111111, 0 = 1111111111

| | |
|---|---|
| Stellenzahl | 10 |
| Bewertbarkeit | Nein |
| Gewicht | 1 ... 10 |
| Minimaldistanz | 1 |
| Maximaldistanz | 9 |
| Hammingdistanz | 1 |
| Stetig | Ja |
| Redundanz | 6,68 |

## 1 aus 10 Code

- 1 = 000000000**1**; 2 = 00000000**1**0; 3 = 0000000**1**00; ...; 9 = 0**1**00000000; 0 = **1**000000000

| | |
|---|---|
| Stellenzahl | 10 |
| Bewertbarkeit | Ja |
| Gewicht | 1 |
| Minimaldistanz | 2 |
| Maximaldistanz | 2 |
| Hammingdistanz | 2 |
| Stetig | Ja |
| Redundanz | 6,68 |

## Gray-Code

- Einschrittiger Code. Wird bei Codelinealen/Codescheiben verwendet.
- Von Codewort zu Codewort ändert sich immer genau eine Stelle.
- 4-Bit Graycode:
  d0001; 0011; 0010; 0110; 0111; 0101; 0100; 1100; 1101; 1111; 1110; 1010; 1011; 1001; 1000

| | |
|---|---|
| Stellenzahl | Unendlich |
| Bewertbarkeit | Nein |
| Gewicht | Unendlich |
| Minimaldistanz | 1 |
| Maximaldistanz | 1 |
| Hammingdistanz | 1 |
| Stetig | Ja |
| Redundanz | 0 |

# 4.3 Alphanumerische Codes

## 4.3.1 ASCII-Code

- American Standard Code for Information Interchange
- Mit 8 Bit = 1 Byte werden alphanumerische Zeichen kodiert.
- Es könnten $2^8$ = 256 verschiedene Zeichen dargestellt werden.
- Es werden jedoch nur 7 Bit verwendet. (128 Zeichen).
- Das erste Bit (Prüfbit) ist 1, wenn die Quersumme der Bit mit dem Wert 1 geradzahlig ist.

### ASCII-Tabelle

| Dez | | 0 | 16 | 32 | 48 | 64 | 80 | 96 | 112 |
|---|---|---|---|---|---|---|---|---|---|
| | Hex | 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 |
| 0 | 0 | NUL | DLE | (Blank) | 0 | @ | P | ' | p |
| 1 | 1 | SOH | DC1 | ! | 1 | A | Q | a | q |
| 2 | 2 | STX | DC2 | " | 2 | B | R | b | r |
| 3 | 3 | ETX | DC3 | # | 3 | C | S | c | s |
| 4 | 4 | EOT | DC4 | $ | 4 | D | T | d | t |
| 5 | 5 | ENQ | NAK | % | 5 | E | U | e | u |
| 6 | 6 | ACK | SYN | & | 6 | F | V | f | v |
| 7 | 7 | BEL | ETB | ‘ | 7 | G | W | g | w |
| 8 | 8 | BS | CAN | ( | 8 | H | X | h | x |
| 9 | 9 | HT | EM | ) | 9 | I | Y | i | y |
| 10 | A | NL | SUB | * | : | J | Z | j | z |
| 11 | B | VT | ESC | + | ; | K | [ | k | { |
| 12 | C | NP | FS | , | < | L | \ | l | \| |
| 13 | D | CR | GS | - | = | M | ] | m | } |
| 14 | E | SO | RS | . | > | N | ^ | n | ~ |
| 15 | F | SI | US | / | ? | O | _ | o | (DEL) |

## 4.3.2 ANSI-Zeichensatz

- Da beim ASCII-Zeichensatz also 128 Zeichen noch frei sind, hat man neue Zeichensätze definiert.
- Diese sind von Standard zu Standard anders. Die ersten 128 Zeichen sind jedoch immer gleich!
- In Westeuropa findet man den ISO-8859-1 Zeichensatz der auch ISO-Latin-1 Zeichensatz genannt wird.
- Es gibt noch andere ISO-8859-Zeichensätze! In Westeuropa wird jedoch der ISO-8859-1-Zeichensatz verwendet!

### 4.3.3 Unicode

- Internationaler Standard für die Zeichen aller Schriften der Welt.
- Alphabete: Lateinisch, Arabisch, indisch; Geografische Zeichen: Chinesisch; Symbole: Mathematik, Technik
- Löst ASCII (7-Bit) und ISO-Latin-1 (8-Bit) ab.
- Die ersten 128 Zeichen sind die vom ASCII-Code; die ersten 256 Zeichen sind vom ISO-Latin-1
- Unicode ist 16-Bit: $2^{16}$ = 65536 mögliche Zeichen.

## 4.4 Strichcodes

- Texterkennung (OCR) dauert viel zu lange und ist fehleranfällig
- Man wollte einen „Computer-Code“ mit 1 und 0 (Schwarz oder Weiss)
- EAN (Europäische Artikel Nummerierung) wurde eingeführt mit weissen und schwarzen Balken.
- Wird sehr häufig eingesetzt: Spedition, Supermarkt, Gesundheitswesen, Geräte-Seriennummern
- Abhängig vom gewählten Barcode können Ziffern, Buchstaben und Sonderzeichen dargestellt werden.
- Es gibt über 200 verschiedene Barcodes.

> Ein Strichcode ist eine in Balken und Abständen gespeicherte maschinenlesbare Information.

### 4.4.1 Der EAN-Code

- Nach der Einführung hatte er gleich den Durchbruch und wurde International. Sollte also Internationale Artikel Nummerierung heissen.
- Der EAN-13-Standard hat 13 Ziffern
- Jede handelsübliche Mengen- oder Verpackungseinheit erhält vom Hersteller eine Artikelnummer.
- Verschiedene Länder haben verschiedene Ziffern: CH: 76; DE: 400 bis 440; IT 80 bis 83; FR: 30 bis 37; ÷: 90 und 91; USA/Kanada: 0 bis 09
- Jeder Hersteller erhält vom eigenen Land eine Betriebsnummer
- Lesefehler werden durch eine Prüfziffer erkannt.
- Fehler durch: Zerknitterung, Verschmutzung, falsche Stromimpulse, elektrische Felder
- Sicherheit: Lesefehler müssen automatisch erkannt werden.

### Aufbau EAN-13-Code (13 Ziffern)

| XX | XXXXX | XXXXX | X |
|---|---|---|---|
| Ländererkennung des Produktionslandes | Betriebsnummer des Herstellers | Artikelnummer, welche der Hersteller vergibt | Welche hilft, Lesefehler zu entdecken |

### Die schwarzen Balken

1. Startzeichen: Schwarz – Weiss – Schwarz (Balken haben Standardbreite). Etwas längere Linien.
2. Linker Block: 6 Zahlen codiert mit Code A und B
3. Zwischenblock: Schwarz – Weiss – Schwarz (Balken haben Standardbreite). Etwas längere Linien.
4. Rechter Block: 6 Zahlen codiert mit Code C


5. Endzeichen: Schwarz – Weiss – Schwarz (Balken haben Standardbreite). Etwas längere Linien.

### Prüfziffer des EAN-13 Codes

1. [„Summe ungerader Zahlen“ + 3 * („Summe gerader Zahlen“)] / 10 = Rest X
2. 10 minus Rest X = Prüfziffer

## 4.5 Codesicherung

- Bei Übertragung können Fehler vorkommen
- Angabe über Fehlerhäufigkeit sagt die Bitfehlerwahrscheinlichkeit von z.B. 0,000001 aus.

> Aufgabe der Codierung ist es, Fehler zu erkennen und zu beseitigen

- Fehler sind erkennbar, wenn Codeworte vorkommen, die nicht im Codewort-Vorrat definiert sind.
- Wir benötigen eine Redundanz, damit wir über ungültige Codewörter verfügen.

## 4.6 Feherlerkennende Codes

Seite 21 im Block 5.

## 4.7 Fehlerkorrigierende Codes

# 5 Dateiformate und Dateistrukturen

## 5.1 Die Datei

- Datei: Struktur, mit der man verschiedenartige und umfangreichere Daten zusammenfassen kann
- Eine Datei kann z.B. Text enthalten oder Informationen zum Bildaufbau.
- Eine Datei hat grundsätzlich einen Namen und sehr oft eine Erweiterung (File-Extension), damit man weiss was in der Datei ist und um was für eine Datei es sich handelt.
- Eine Datei ist eine lange Folge von Bytes.

### 5.1.1 Dateiattribute

- Zusätzlich zur eigentlichen Datei werden weitere Informationen gespeichert.
- Erstelldatum, Bearbeitungsdatum, letzter Zugriff, wer erstellte die Datei, wer darf Datei ändern, Anmerkungen, mit welchem Programm wird gestartet, etc.

### 5.1.2 Dateistrukturen

- Eine Datei ist eine lange Folge von Bits.
- Oft wird 1Byte = 8 Bit als elementares Teilchen einer Datei betrachtet.
- Dateien sind nach bestimmten Regeln aufgebaut.
- Häufig mit einem Header (Beschreibung im Kopf) und mit einem Body (Inhalt).

### 5.1.3 Dateiformat-Einteilung

- Dateien werden gebraucht für: Archivierung, Systemsteuerung, Coderiung, Dokument- und Texterstellung, Audiodateien, Grafikdateien, Videodateien, Textdateien, etc...

### 5.1.4 PDF-Dateien

- Löste mehr und mehr RTF (Rich Text Format) ab.
- Betriebssystemunabhängig: Sieht auf jedem Rechner exakt gleich aus. Layout immer gleich.
- Wurde von Adobe erfunden.
- Schriften können eingebettet werden.
- Jeder kann gratis PDF erstellen und lesen. Oft wird durch ein PDF-Drucker ein PDF erstellt.

# 5.2 Computer-Grafiken

- Computer-Grafiken werden in Dateien gespeichert.

## 5.2.1 Zeichen-, Pixel-, Vektorgrafik

### Zeichengrafik

- Relikt von früher, als auf dem Bildschirm und Drucker nur Textzeichen ausgegeben werden konnte.
- Bild wird mit Hilfe von Textzeichen vom ASCII-Code dargestellt.
- Findet man in Grossrechnern.
- Sehr schnell aus Drucker ausgegeben
- Diese ASCII-Arts werden in der Warez-Szene sehr aktiv angewendet und beworben. Es findet dort ein richtiger Wettbewerb unter den Erstellern statt.
- Der Host der Suva liefert auch solche „Bilder“.

### Pixelgrafik

- Bild besteht aus Bildpunkten (Pixel)
- Computer und Fernseher (768 x 576) können grundsätzlich nur Pixelgrafiken darstellen.
- Jeder Pixel besteht aus drei Elementen: Rot, Grün, Blau (RGB)
- Speicherbedarf: Anzahl der Bildpunkte horizontal und vertikal, Farbe der einzelnen Pixel
  - 640 x 480 mit Grautönen (256 Farben) = 640 * 480 * 1 (1Byte = 8 Bit; $2^8$ = 256) = 307200 Byte
- Anwendung: Schnell auf dem Bildschirm dargestellt. Schnelle Bildfolge = Filmsequenz.
- OCR-Umwandlung möglich.
- Problem: Qualitätsverlust bei Vergrösserung

### Vektorgrafik

- Keine Punkte sondern Bildelemente, welche genau beschrieben sind (Farbe, Dicke, Muster, Lage).
- Durch festgelegte Sprache beschrieben (z.B. SVG)
- Vergrössern ohne Qualitätsverlust
- Bildelemente einzeln manipulierbar
- Bei Ausgabe an Bildschirm oder Drucker wird die Vektorgrafik in eine Pixelgrafik umgewandelt
- Stiftplotter kann Vektorgrafiken als solche Ausgeben.

## 5.2.2 Grafikformate

### Pixelgrafiken

- BMP (Bitmap-Format); unter Windows. Fast alle Farbtiefen. Es gibt auch Varianten mit Kompression, die jedoch kaum zu gebrauchen ist.
- TIFF (Tagged Image File Format): zur Verminderung des Platzbedarfs. Graustufen eingescannte Bilder. Wichtigen Standard zum Austausch von Bildern. Besonders in der Druckindustrie.
- GIF (Graphics Interchange Format): vom Onlinedienst CompuServe entwickelt und wird im Internet viel verwendet. Hohe Kompressionsrate und ist auf 256 Farben begrenzt. Kompression erfolgt allerdings zeilenweise.
- JPEG (Joint Photographic Export Group): Im Internet weit verbreitet. Datenverlust behaftete Kompression, die einstellbar ist.
- PNG (Portable Network Grahic): vom W3C entwickelt und als Standard verabschiedet. Lizenzfrei. Gute Darstellungsqualität. Starke, verlustfreie Komprimierung.
- PCX (PC-Paintbrush-Format): für pixelorientierte Mal- und Zeichenprogramme. Kann von fast allen Grafikprogrammen gelesen und ausgegeben werden.
- PICT (Macintos Format): Wird häufig auf Apple-Rechner verwendet, es könne jedoch auch die meisten Grafikprogramme unter Windows verarbeiten.
- Kodak-Foto-CD: Pixelformat bis zu 3072x2048 Bildpunkten und hoher Farbtiefe. Wurde entwickelt, um Fotografien im Kleinbildformat auf CD-ROM zu speichern. Sie können somit betrachtet und bearbeitet werden. Die Grafiken sind komprimiert. Für die schnelle Suche werden die Bilder zusätzlich mit deutlich geringerer Auflösung gespeichert.
- Flashpix: Es werden kleinere Kopien in diesem Bild mit gespeichert für z.B. eine Vorschau. Daher kann man diese wesentlich schneller anzeigen. Man kann auch komprimieren. Mit einer Steuersprache kann man z.B. Das Bild drehen oder die Farbe verändern, wobei das Original nicht angetastet wird. Diese „Änderungsbeschreibung“ wird auch im Bild gespeichert. Damit man das Bild schneller bearbeiten kann, wird nur immer der aktuelle Bereich, den man auf dem Monitor sieht angezeigt. Diese Teile nennt man Tiles oder auch Kacheln.

### Vektorgrafiken

- EPS (Encapsulated PostScript): wichtigstes Format für Vektorgrafiken. Für Darstellung muss man über PostScript-Intepreter verfügen). Im Image-Header ist ein Vorschaubild gespeichert. Kann aber auch hochauflösende Picelgrafik als Objekt aufnehmen.
- WMF (Windows Meta File): von Windows. Wird vor allem für Windows verwendet. Tjano.
- SVG (Scalable Vector Graphics): Standard für zweidimensionale Vektorgrafiken. Vom W3C empfohlen. XML-Basierend. Einfach schön!
- HPGL (Hewlett Packard Graphics Language): Seitenbeschreibungssprache vor allem für Stiftplotter. Kann aber auch von anderen Ausgabegeräten verwendet werden.
- DXF (Data Exchange Format, Autocad-Format): Wurde durch Autocad zum Standard im CAD-Bereich (Computer Aided Design). Kann von den meisten Rendering-Programme gelesen werden und von vielen Zeichenprogrammen.
- CGM (Computer Graphics Metafile): Wird häufig für Cliparts verwendet und ist weit verbreitet.

# 5.3 Farben

## 5.3.1 Farbtiefe

- Anzahl der Bits, die für die Speicherung der Farbe eines Pixels verwendet werden.
- 24 Bit = $2^{24}$ = 16,7 Mio Farben
- Je höher die Farbtiefe, desto feiner der Unterschied zwischen zwei nebeneinanderliegenden Farben

## 5.3.2 RGB-Farbmodell / Additives Farbmodell

- Die drei Grundfarben Rot Grün Blau werden mit unterschiedlicher Intensität übereinandergelegt.
- Alle Farben = weiss; keine Farbe = schwarz

## 5.3.3 CMYK-Farbmodell / Subtraktives Farbmodell

- Zum Drucken, da der Drucker nicht mit allen Farben weiss mischen kann.
- Man braucht die Komplementärfarben zu RGB:
  - Rot: Cyan; Grün: Magenta; Blau: Yellow
- Vierte Farbe ist schwarz, da man schwarz schlecht mischen kann :-)

# 5.4 Speicherplatzbedarf bei Fotos

### Tabelle

| Bildgrösse | Farbtiefe | Farbanzahl | Speicherbedarf | Bemerkung |
|---|---|---|---|---|
| 640x480 | 1 Bit | 2 | 38 KB | |
| 640x480 | 4 Bit | 16 | 150 KB | VGA |
| 640x480 | 24 Bit | 16,7 Mio | 900 KB | |
| 800x600 | 8 Bit | 256 | 469 KB | Super VGA |
| 800x600 | 24 Bit | 16,7 Mio | 1,4 MB | |
| 1024x768 | 24 Bit | 16,7 Mio | 2,3 MB | |
| 1280,1024 | 24 Bit | 16,7 Mio | 3,8 MB | |
| 1600x1024 | 24 Bit | 16,7 Mio | 5,6 MB | |
| 3072x2048 | 24 Bit | 16,7 Mio | 18,4 MB | Photo-CD |
| 6144x4096 | 24 Bit | 16,7 Mio | 73,7 MB | Pro Photo-CD |

### Berechnen

Farbtiefe: Rot = 8 Bit; Grün = 8 Bit, Blau = 8 Bit; --> 3 * 8 Bit = 24 Bit

$Speicherbedarf_{Unkomprimiert}$ = Farbtiefe * Anzahl_Pixel (experimentell!!!)

# 6 Datenkompression

## 6.1 Komprimierung

- Viele Programme verwenden die Huffman-Komprimierung (z.B. Das ZIP-Format)
- Huffman Komprimiert ohne Verlust. Die komprimierte Datei hat etwa noch 1/3 der Originalgrösse.
- Komprimierte Daten können nicht noch einmal komprimiert werden. Es bringt also nichts, die Fotosammlung oder die MP3-Sammlung zu komprimieren!
- ZIP ist ideal für E-Mail.
- Mit Passwort versehbar, jedoch nicht sehr sicher.
- Selbstextrahierende Archive liefern ein Entpackungsprogramm in der Datei mit. Somit braucht der Empfänger kein Entpackungsprogramm.
- Die meisten Komprimierungen verwenden heute noch die Huffman Kodierung
- Dateien haben oft hohen Grad an Redundanz und somit geringen Informationsgehalt.
- Redundanzen sollten eliminiert werden.
- Komprimierung kann auf alle Dateien angewandt werden. Bei schon komprimierten Dateien hilft es nur wenig.
- Dateien mit zufälligen Bits sind oft nicht komprimierbar.
- Bei kleinen Dateien kann eine Komprimierung die Datei sogar verlängern.
- Komprimiete Dateien sind sensitiv gegenüber Fehler
    - Es empfiehlt sich eine gewisse Redundanz einzuführen
    - Die moderne Tendenz beseteht darin, Komprimierung und Sicherheit von einander zu trennen und separat zu lösen.

## 6.2 Lauflängenkodierung

- AAAAABBBBBBBCCC kann man in 5A7B3C umwandeln
- Und schon ist es kürzer
- Die Wiederherstellung ist schnell gemacht.
- Bei binären Zeichenfolgen kann zusätzlich der Wert der Zeichen eingespart werden, weil sie alternierend aus 0 und 1 bestehen.
- Nur das erste Zeichen muss bekannt sein
- Erfordert die Darstellung von Zählern und Zeichen: z.B. 3*A, 5*W.
    - Um dieses Problem zu umgehen, kann ein Escape-Zeichen verwendet werden, das eine Sequenz Zähler, Zeichen ankündigt, z.B. Q = Escape-Zeichen: QDABBAAQEBQHCDABCBAA

### Beispiel zur Lauflängenkodierung

| Binäres Bild | Lauflängenkodierung |
|---|---|
| 00000**1**0000<br>0000**11**0000<br>000**1111**000<br>00**111111**00<br>0**111111111**0 | 5 1 4<br>4 2 4<br>3 4 3<br>2 6 2<br>1 8 1 |

- Originalgrösse = 10 Spalten * 5 Zeilen * 1 Bit = 50 Bit
- Komprimierte Grösse: Es sind 10 Zeichen pro Zeile. Also brauche ich maximal 4 Bit, weil $2^4 = 16$. Somit braucht man für die binäre Speicherung des Bildes mit der Lauflängenkodierung 15 Lauflängenangaben * 4 Bit/Längenangabe = 60 Bit.
  In diesem Fall hat sich die Grösse der Datei sogar vergrössert!

## 6.3 Kodierung variabler Länge

- Wenn in einem Text 1000 mal ein „A“ vorkommt, braucht jeder einzelne Buchstabe genau gleich viel Platz wie der einzige „W“. Das ist bei der Speicherung nicht sehr effizient.
- Je häufiger ein Zeichen im Text vorkommt, desto kürzer soll sein Code sein.

### Beispiel

Fixe Länge: ADAM (A = 0000, D = 0001, M = 0010) = 4 * 11 Bits = 44 Bits

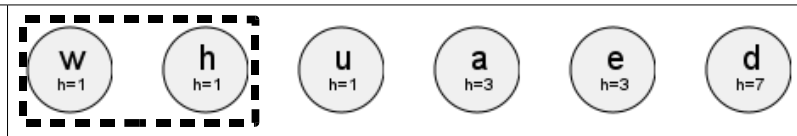
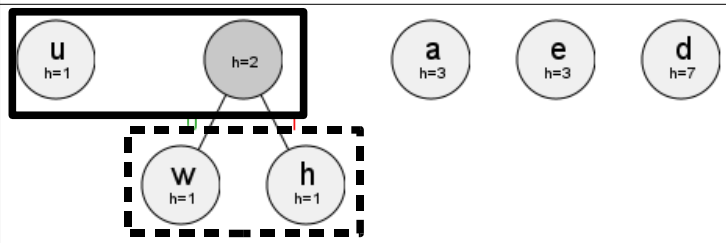
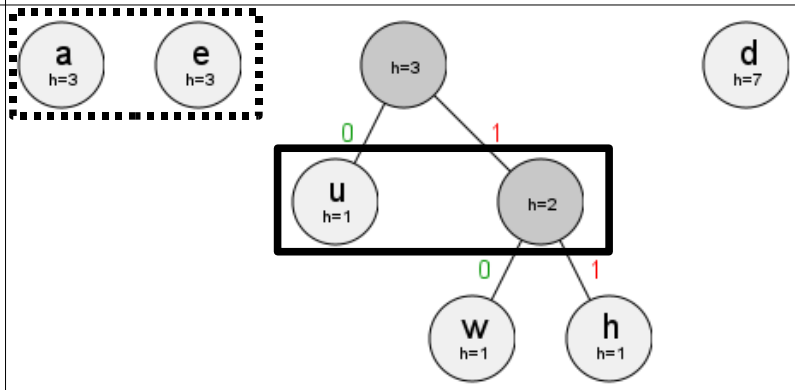
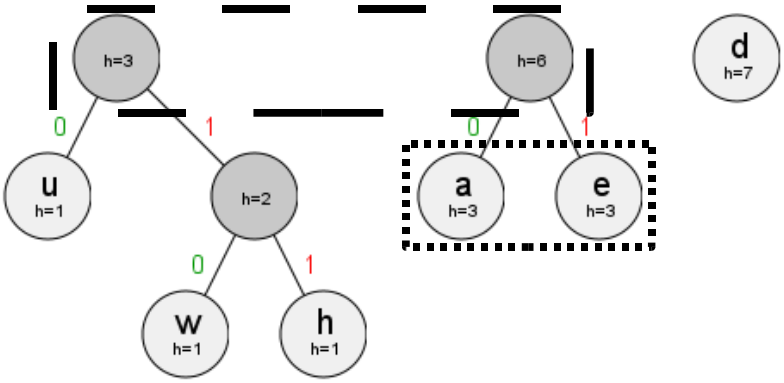
Variable Länge: ADAM (A = 01, D = 110, M = 101) = 10 Bits.

## 6.4 Die Huffman-Kodierung

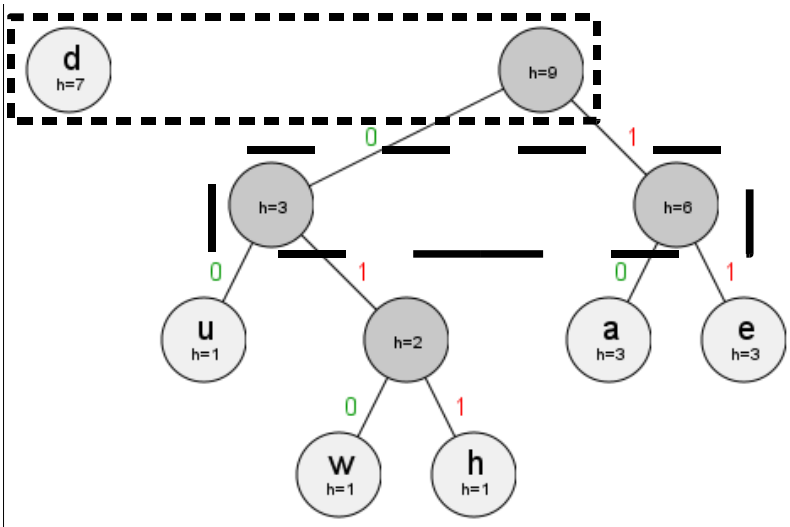
- Kodierung mit variabler Länge.
- Ziel ist es die Länge der häufig auftretenden Zeichen zu minimieren.
- MPEG, Zip brauchen dieses Verfahren.
- Kein anderer Zeichencode führt zu einer kürzeren Kodierung als die Huffman-Kodierung.
- Huffman löste das Problem, wo ein Buchstabe zu ende ist.
- Zu Beginn der Huffman-Codierung werden aus den zu codierenden Zeichen Bäume erstellt und diese ihrer Häufigkeit nach aufsteigend in einem Wald sortiert.
  - Die Erstellung des Codebaums, aus dem später der Code abgeleitet wird, geschieht in zwei Schritten. Zuerst werden die beiden Bäume mit den kleinsten Häufigkeiten zu einem neuen Baum zusammengefasst. Danach wird dieser Baum wieder in den Wald einsortiert.
  - Diese beiden Schritte werden solange wiederholt, bis nur noch ein Baum vorhanden ist.
- Der Codierbaum wird in der Datei gespeichert. Dieser braucht auch Platz! Statt des Baumes kann man auch die Zeichenhäufigkeit speichern, wenn Encoder und Decoder genau den selben Baum erzeugen.
- Kommen in einer Datei alle Zeichen gleich oft vor, funktioniert es nicht.

- Die Quelldatei muss zweimal eingelesen werden: Einmal für den Baum zu erstellen. Das andere mal für die eigentliche Codierung. Daher ist es keine Live-Kodierung.
- Super Tool zum Üben: http://www.tillwiebke.de/tools/hsf/

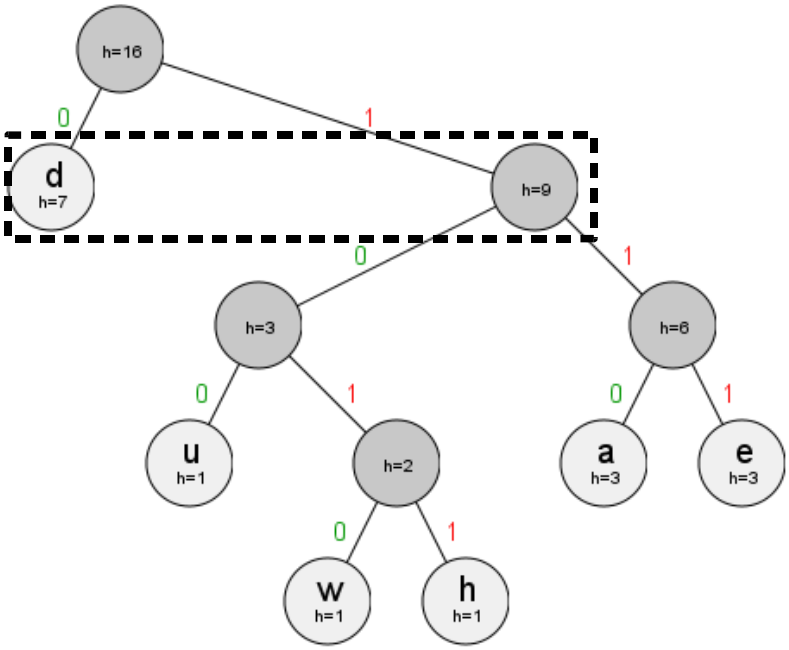
## Beispiel mit „waddehaddedudeda“

| | |
|---|---|
| Buchstaben auflisten. Häufigkeit angeben uns danach sortieren. |  |
| Der aus den Bäumen "w" und "h" neu erstellte Baum mit der Häufigkeit 2 wurde zwischen den Bäumen "u" (h=1) und "a" (h=3) eingefügt. |  |
| Der aus den Bäumen "u" und "wh" neu erstellte Baum mit der Häufigkeit 3 wurde zwischen den Bäumen "e" (h=3) und "d" (h=7) eingefügt. |  |
| Der aus den Bäumen "a" und "e" neu erstellte Baum mit der Häufigkeit 6 wurde zwischen den Bäumen "uwh" (h=3) und "d" (h=7) eingefügt. |  |

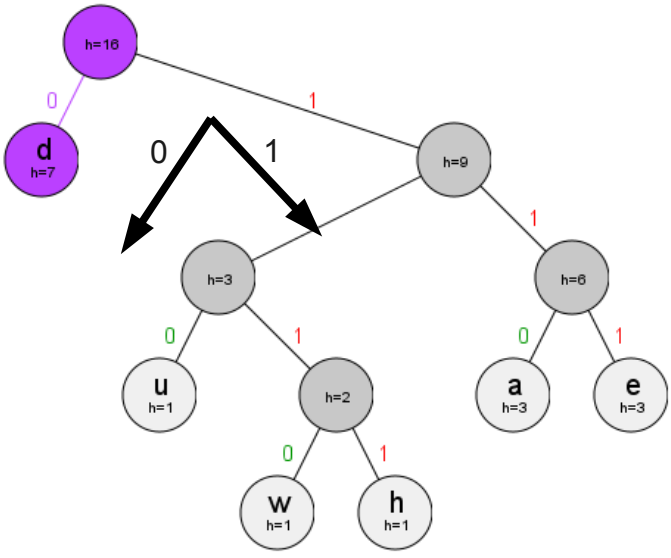
Der aus den Bäumen "uwh" und "ae" neu erstellte Baum mit der Häufigkeit 9 hatte die gröflte Häufigkeit, weswegen er am Ende des Waldes angehängt wurde.




Der aus den Bäumen "d" und "uwhae" neu erstellte Baum mit der Häufigkeit 16 ist der einzige Baum des Waldes. Nun kann mit der Erstellung der Codes für die einzelnen Zeichen begonnen werden.




Nach der Erstellung des Baumes wird aus ihm die Codierung abgeleitet. Dazu werden die Kantenbezeichnungen von der Wurzel zu den Blättern zusammengefasst und dem im Blatt gespeicherten Zeichen zugeordnet.



Die Erstellung des Codes ist abgeschlossen. Sie können die Codierung der einzelnen Zeichen der Tabelle oben rechts entnehmen.
Zur Codierung eines Zeichens werden durchschnittlich 2.25 Bits verwendet.

| Zeichen | Häufigkeit | Code |
|---|---|---|
| d | 7 | 0 |
| a | 3 | |
| e | 3 | |
| w | 1 | |
| h | 1 | |
| u | 1 | |

### Notizen dazu

1. Nach Häufigkeit alphabetisch sortieren.
2. Von Links: die ersten beiden nehmen und zusammenfassen.
3. Nach der Häufigkeit so weit rechts wie möglich einsortieren.
4. Die nächsten beiden ersten nehmen, usw. Bis man nur noch 1 Baum hat.
5. Nach rechts = 1; nach links = 0
6. Codierungstabelle schreiben

## 6.4.1 Dynamische Huffman-Kodierung

- Der Baum wird während der Codierung der Datei angelegt. Bei jedem codierten Byte wird der Baum angepasst.
- Die Kompressionsrate ist dann geringer, weil sich der Packer an die Daten „gewöhnen“ muss.

# 7 Verschlüsselung

## 7.1 Kryptografie

- Informationen sollen unkenntlich gemacht werden
- Es sollen keine Rückschlüsse auf den Inhalt gemacht werden können.

### 7.1.1 Ziele der Kryptographie

Die Kryptographie hat vier Hauptziele:

- Vertraulichkeit der Nachricht
- Datenintegrität
- Authentifizierung
- Verbindlichkeit

### 7.1.2 Chronologie der Kryptographie

**Kryptografie** bzw. **Kryptographie** (vom griechischen *kryptós*, „verborgen“, und *gráphein*, „schreiben“) ist im ursprünglichen Sinne die Wissenschaft der Verschlüsselung von Informationen („Geheimschriften“). Heutzutage beschäftigt sie sich allgemein mit dem Schutz von Daten durch deren Transformation, in der Regel unter Einbeziehung von geheimen Schlüsseln. Die Kryptografie bildet mit der Kryptoanalyse zusammen die Kryptologie.

Obwohl die Kryptografie eine lange und komplexe Geschichte hat, entwickelte sie sich erst im 20. Jahrhundert zur rigorosen und auf Mathematik basierenden Wissenschaftsdisziplin. Aber erst mit den Kommunikationsmöglichkeiten des Internets kam sie in den allgemeinen, jedermann zugänglichen Gebrauch.

### Monoalphabetische Substitution – Caesar Code

Bei dieser besonders simplen Variante einer einfachen monoalphabetischen Substitution wird das zur Verschlüsselung verwendete Alphabet durch zyklisches Verschieben jedes einzelnen Buchstaben des Standardalphabets gewonnen (siehe auch: Verschiebechiffre). Die Anzahl der Plätze, um die verschoben wird, ist der Schlüssel. Caesar selbst benutzte stets den Schlüssel 3, also die im folgenden Beispiel illustrierte Verschlüsselung (die für die damaligen gallischen und germanischen Kryptanalysten vermutlich „unknackbar“ war).

### Polyalphabetische Substitution – Vigenère-Code

Die Vigenère-Verschlüsselung (nach Blaise de Vigenère) galt lange als sicherer Chiffrieralgorithmus. Ein Schlüsselwort bestimmt, wie viele Alphabete genutzt werden. Die Alphabete leiten sich aus der Caesar-Substitution ab.
Dem britischen Mathematiker Charles Babbage gelang um das Jahr 1854 erstmals die Entzifferung einer Vigenère-Chiffre. Diese Entdeckung wurde jedoch damals nicht öffentlich bekannt gemacht. Der preuflische Offizier Friedrich Kasiski veröffentlichte im Jahr 1863 seine Lösung und ging damit in die Geschichte ein.

# 7.2 Symmetrisches Verschlüsselungsverfahren

- Absender und Empfänger besitzen den gleichen Schlüssel.
- Dieses Verfahren ist sehr schnell.
- Bietet eine gute Sicherheit.
- Das Problem ist der Austausch der Schlüssel.
- Wenn ein dritter diesen Schlüssel kennt, kann er die Daten manipulieren.
- **Substitution**: Einzelne Zeichen eines Textes durch andere tauschen
- **Transposition**: Reihenfolge der Zeichen ändern.

## 7.2.1 Substitution

Bei der Substitution werden die Buchstaben bzw. Wörter nicht in eine andere Reihenfolge gebracht, sondern ersetzt durch andere Zeichen oder Buchstaben.

Bei der Verschlüsselung durch Substitution wird jeder Buchstabe im Klartext durch einen anderen Buchstaben (oder ein anderes Zeichen) ersetzt: Bei der Transposition bleibt sich jeder Buchstabe gleich, doch er wechselt seinen Platz, während bei der Substitution jeder Buchstabe seine Gestalt wechselt, jedoch seinen Platz behält.

Beispiele sind der Freimaurer-Code oder der Code des Polybos.

### Monoalphabetische Substitution – Caesar Code

Bei dieser besonders simplen Variante einer einfachen monoalphabetischen Substitution wird das zur Verschlüsselung verwendete Alphabet durch zyklisches Verschieben jedes einzelnen Buchstaben des Standardalphabets gewonnen (siehe auch: Verschiebechiffre). Die Anzahl der Plätze, um die verschoben wird, ist der Schlüssel. Caesar selbst benutzte stets den Schlüssel 3, also die im folgenden Beispiel illustrierte Verschlüsselung (die für die damaligen gallischen und germanischen Kryptanalysten vermutlich „unknackbar“ war).

### Polyalphabetische Substitution – Vigenère-Code

Die Vigenère-Verschlüsselung (nach Blaise de Vigenère) galt lange als sicherer Chiffrieralgorithmus. Ein Schlüsselwort bestimmt, wie viele Alphabete genutzt werden. Die Alphabete leiten sich aus der Caesar-Substitution ab.

Dem britischen Mathematiker Charles Babbage gelang um das Jahr 1854 erstmals die Entzifferung einer Vigenère-Chiffre. Diese Entdeckung wurde jedoch damals nicht öffentlich bekannt gemacht. Der preuflische Offizier Friedrich Kasiski veröffentlichte im Jahr 1863 seine Lösung und ging damit in die Geschichte ein.

## 7.2.2 Transposition

Bei der Transposition werden die Buchstaben des Klartexts in eine andere Reihenfolge gebracht, es wird gewissermassen ein Anagramm (Buchstabenversetzrätsel) erzeugt. Diese Reihenfolge muss aber dem Empfänger der Nachricht bekannt sein. Ein Beispiel ist hier die spartanische Skytale.

# 7.3 Asymmetrisches Verschlüsselungsverfahren

- Es gibt private und öffentliche Schlüssel.
- Dieses Verfahren nennt man PKI (Public Key Infrastructure)
- Es wird mit dem öffentlichen Schlüssel verschlüsselt.
- Der dazugehörige private Schlüssel kann die Nachricht wieder entschlüsseln.
- Dieser Vorgang ist 1000 mal langsamer als die symmetrische Verschlüsselung.

## 7.3.1 Hybridverfahren

- Die Schlüssel für die symmetrische Verschlüsselung werden mit der asymmetrischen Verschlüsselung übertragen.

## 7.3.2 Authentifikation

- Mit dem privaten Schlüssel wird eine Nachricht verschlüsselt.
- Mit dem dazugehörigen öffentlichen Schlüssel wird die Nachricht entschlüsselt.
- Der öffentliche Schlüssel kann jeder einsehen.
- Die Nachricht kann gelesen, jedoch nicht manipuliert werden, weil man zum erneuten Verschlüsseln wieder den privaten Schlüssel braucht.

## 7.3.3 Elektronische Unterschrift

1. Hash-Wert einer Nachricht wird mit dem privaten Schlüssel verschlüsselt.
2. Empfänger entschlüsselt den Hash-Wert mit dem dazugehörigen öffentlichem Schlüsel.
3. Empfänger bildet auch den Hash-Wert.
4. Ist der Hash-Wert der selbe, dann wurde die Nachricht nicht verändert.

## 7.3.4 Zertifikate

- Damit man sichergehen kann, dass der öffentliche Schlüssel wirklich von der anzunehmenden Person stammt, wurden Zertifizierungsstellen eingerichtet.
- Diese Zertifizierungsstellen veröffentlichen den eigenen öffentlichen Schlüssel in einem Zertifikat, das von der Zertifizierungsstelle elektronisch unterschrieben ist.

## 7.3.5 Kryptoanalyseverfahren

### Kryptoanalyse – Babbage, Kasiski, Friedman

Schwachstelle: Die Kenntnis der Schlüssellänge reicht aus, um einen hinreichend langen Geheimtext zu entschlüsseln.

### Perfekte Verschlüsselung – Vernam, Baudot

Es gibt ein kryptoanalytisches Verfahren, das man ohne Kenntnis des Schlüsselwortes nicht zu entschlüsseln vermag. Die Unknackbarkeit ist (mathematisch) beweisbar – kein Rechner, auch nicht alle Rechner vereint, werden jemals in der Lage sein, einen im Vernam-Baudot-Verfahren verschlüsselten Text zu knacken.

Ganz wichtig dabei ist die Zufälligkeit des Schlüssels. Pseudozufälligkeit macht die Chiffre bereits angreifbar.

### Enigma

Die ENIGMA ist eine Rotor-Schlüsselmaschine, die im Zweiten Weltkrieg im Nachrichtenverkehr des deutschen Militärs verwendet wurde. Auch andere Dienststellen, wie Polizei, Geheimdienste, diplomatische Dienste, SD, SS, Reichspost und Reichsbahn, setzten sie zur geheimen Kommunikation ein. Das Wort „Enigma“ (αίνιγμα) kommt aus dem Griechischen und bedeutet Rätsel.

### DES - Data Encryption Standard

Der DES ist der im kommerziellen Bereich am häufigsten eingesetzte Verschlüsselungsalgorithmus. Wurde von Horst Feistel, einem Deutschen, der 1934 nach Amerika emigriert war, entwickelt.

DES verschlüsselt Nachrichten so, indem er sie systematisch durcheinander bringt. Der Text wird in eine binäre Folge verwandelt und anschliessend in Blöcke à 64 Bits aufgeteilt ▪ Blockchiffre.

### IDEA - International Data Encryption Algorithm

Er wurde an der ETH Zürich entwickelt und ist genauso ein Blockchiffre (auch à 64bits) wie der DES. Der grosse Unterschied ist allerdings, dass die Schlüssellänge 128 Bits beträgt.

## 7.4 Begriffe aus der Kryptologie

- **Kryptographie** ist die Wissenschaft, die mit Hilfe mathematischer Methoden Daten verschlüsselt und wieder entschlüsselt. Das erklärte Ziel ist, sicheren Datenaustausch zu gewährleisten.
- **Kryptoanalyse** ist die Wissenschaft, die sich mit der Analyse von Geheimtexten befasst und das ultimative Ziel ist, diese in Klartexte zu verwandeln. Kryptoanalysten werden auch als Angreifer bezeichnet.
- Der Oberbegriff **Kryptologie** fasst diese beiden Bereiche zusammen.

# 8 Glossar

| | |
|---|---|
| FLOPS | Floating Point per Second |
| MIPS | Million instructions per second |
| Samplingrate | Million instructions per second |
| Interpolation | Auf mathematischem Weg zusätzliche Bildpunkte errechnet, die sich zwischen anderen Bildpunkten befinden. Die Interpolationsverfahren dienen der Erhöhung der scheinbaren Auflösung von Grafiken, Bildern und Fotos. |
| Integrität | Unveränderte Daten, die echt und unverfälscht sind im gesamten als Datei. |
| Authentizität | Überprüfung ob die Nachricht wirklich vom Sender ist und unverändert ist. |

# 9 Gute Links

- **Zahlensysteme**: http://www.arndt-bruenner.de/mathe/scripts/Zahlensysteme.htm
-

# Stichwortverzeichnis