ZUSAMMENFASSUNG – MATHEMATIK VERTIEFUNG

Kreienbühl Mika – Studierender Techniker HF System- & Netzwerktechnik mit Cyber-Security

Mika Kreienbühl
mika.kreienbuehl@student.ipso.ch

Inhalt

Übersicht Mathematik Vertiefung

Im Rahmen der Techniker HF an der IFA (ipso Bildung) werden im 3. Semester Mathematik Vertiefungen erarbeitet, welche im Laufe der weiteren Weiterbildung wichtig für das Verständnis der Funktion vom Computer und dessen Algorithmen sind.

Als Referenz dieser Zusammenfassung dienen die Unterrichtmaterialien, welche im Rahmen der Techniker HF im Herbst/Winter Semester 2021/22 an der IFA ausgehändigt wurden. Ausserdem werden die Notizen sowie selbst erarbeitete Informationen verwendet.

Sämtliche Angaben in diesem Dokument sind ohne Gewähr und jegliche Haftung wird abgelehnt.

Fächerinfos

Schule	IFA, Bern
Lehrgang	Techniker HF System- & Netzwerktechnik mit Cyber-Security
Dozierender	Senol Sakru
Unterrichtszeitraum	11.2022 bis 12.2022

Zahlensysteme

Es gibt mehrere Zahlensysteme welche in der Informatik sowie Mathematik zum Einsatz kommen. Das gängigste Zahlensystem ist das Dezimalsystem. Dies beinhaltet die alte bekannten Ziffern 0 bis 9. Wobei es weitere gängige Zahlensysteme wie Binär, Okta und Hexadezimal gibt. Das Oktasystem wird nicht weiter erläutert.
In der Informatik werden – um Verwechslungen zu vermeiden – die Zahlen mit einem Index versehen, der definiert, um welches Zahlensystem es sich handelt:

Mathematik	1001_2	1001_{10}	1001_{16}
Informatik	0b1001	0d1001	Ox1001

Dezimalsystem

Das Dezimalsystem verwendet die Ziffern 0 bis 9.

10^6	10^5	10^4	10^3	10^2	10^1	10^0
1’000’000	100’000	10’000	1000	100	10	1
1	2	5	7	3	8	5
In Dezimal: 1’257’385						

Binärsystem

Das Binärsystem verwendet die Ziffern 0 und 1

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	1	1	0	0	1	1	1
In Dezimal: 103							

Umrechnung

Binär zu Dezimal

01100111	
Rechnung	**Ergebnis**
0 * 128	0
1 * 64	64
1 * 32	32
0 * 16	0
0 * 8	0
1 * 4	4
1 * 2	2
1 * 1	1
	= 103

Dezimal zu Binär

103		
Rechnung	**Ergebnis**	**Binärgewicht**
103 : 2 = 51 Rest 1	1	2^0
51 : 2 = 25 Rest 1	1	2^1
25 : 2 = 12 Rest 1	1	2^2
12 : 2 = 6 Rest 0	0	2^3
6 : 2 = 3 Rest 0	0	2^4
3 : 2 = 1 Rest 1	1	2^5
1 : 2 = 0 Rest 1	1	2^6
1100111		

Hexadezimalsystem

Das Hexadezimalsystem verwendet die Ziffern 0-F (Wert: 15).

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

16^9	16^8	16^7	16^6	16^5	16^4	16^3	16^2	16^1	16^0
…	…	…	…	1'048’576	65’536	4’096	256	16	1
-	-	-	-	-	-	A	C	D	C
-	-	-	-	-	-	40’960	3’072	208	12
In Dezimal: 44’252									

Umrechnung

Hexadezimal zu Dezimal

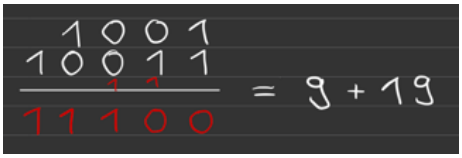
ACDC			
Hexadezimalwert	**Dezimalwert**	**Rechnung**	**Ergebnis**
C	12	12 * 1	12
D	12	12 * 16	208
C	12	12 * 256	3’072
A	10	12 * 4096	40’960
			= 44’252

Dezimal zu Hexadezimal

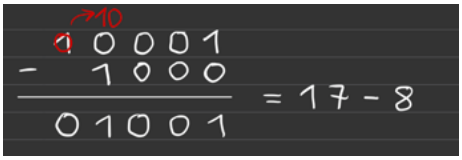
44’252		
Rechnung	**Ergebnis**	**Hexagewicht**
44'252 : 16 = 2765 Rest 12	12	C
2765 : 16 = 172 Rest 13	13	D
172 : 16 = 10 Rest 12	12	C
10 : 16 = 0 Rest 10	10	A
ACDC		

Rechnen in Binär

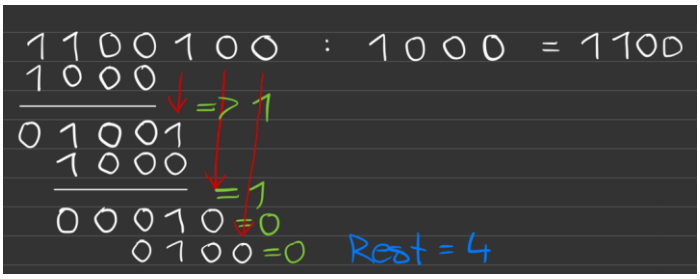
Addition



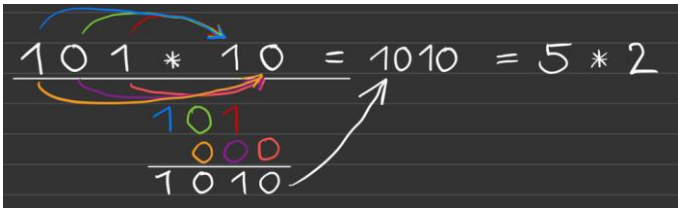
Subtraktion



Division



Multiplikation



Man beginnt bei der Multiplikation bei der letzten Ziffer der ersten Zahl und bei der ersten Ziffer der zweiten Zahl. Anschliessend der zweiten Ziffer der ersten Zahl und der ersten Ziffer der ersten Zahl. Dies, bis man bei der ersten Ziffer der ersten Zahl angekommen ist. Anschliessend «arbeitet» man die zweite Ziffer der zweiten Zahl wiederum nach dem selben Schema ab.

Logische Schaltungen

AND-Verknüpfung

Damit diese logische Verknüpfung den Binärwert 1 ausgibt, müssen beide Inputs den Binärwert 1 haben.

Symbol: & 

A	B	Ausdruck (A&B)
1	1	1
1	0	0
0	1	0
0	0	0

OR-Verknüpfung

Der Verknüpfung gibt den Binärwert 1 aus, wenn mindestens eine der beiden Inputs den Binärwert 1 hat.

Symbol: || 

A	B	Ausdruck (A||B)
1	1	1
1	0	1
0	1	1
0	0	0

XOR-Verknüpfung

Die Verknüpfung gibt 1 aus, wenn nur eine der Inputs den Binärwert 1 hat.

Symbol: # 

A	B	Ausdruck (A#B)
1	1	0
1	0	1
0	1	1
0	0	0

NOT-Verknüpfung

Die Verknüpfung gibt 1 aus, wenn der Input 0 ist.

Symbol: ! 

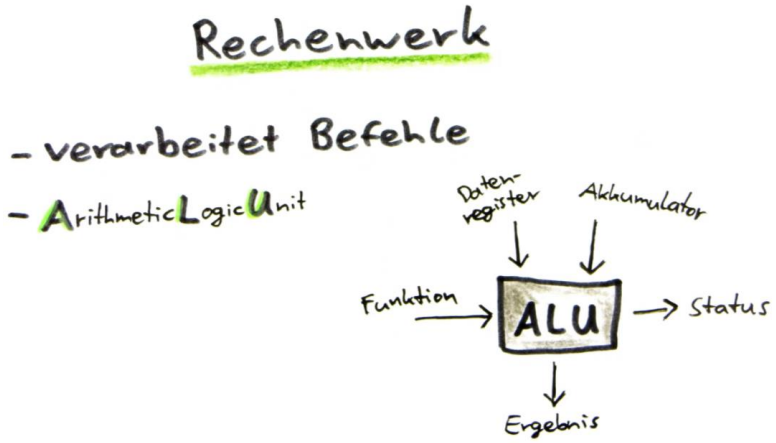
A	!A
0	1
1	0

NAND-Verknüpfung

NAND sagt aus, dass alle Zustände den Binärwert 1 ausgeben, ausser es haben beide Inputs den Wert
Symbol: !(A&B) 

A	B	Ausdruck !(A&B)
1	1	0
1	0	1
0	1	1
0	0	1

ALU (Aritmetic Logical Unit)



Codierungstypen

Codierung	Bits
ASCII	7 Bit
ANSI	8 Bit
Unicode	16 Bit

Negative Werte

Zweierkomplement

Das Zweierkomplement löst das Problem, dass der Computer keine negativen Zahlen darstellen kann. Durch das Zweierkomplement werden die Werte 0 bis 127 normal interpretiert. Die Gegenzahl erreicht man durch das Invertieren aller Bits und anschliessender Addition von 1. Diese Methode wird für Operationen auf den ALU verwendet. Ist jedoch nicht für Berechnungen geeignet.

Binärzahl	**Wert normal**	**Wert im Zweiterkomplement**
0000 0000	0	0
0000 0001	1	1
0000 0010	2	2
…		
0111 1111	127	127
1000 0000	128	-128 (-256)
…		
1111 1110	254	-2
1111 1111	255	-1

Exzess-Codierung

Auch die Exzess-Codierung löst das Problem mit der negativen Zahlendarstellung.
Man kann die negativen Zahlen darstellen, indem 127 plus den darzustellenden Wert gerechnet wird. Es wird für die Darstellung von Gleitkommazahlen genutzt.

Fehlererkennung

Paritätsbit

Beim Paritätsbit wird das Bit zur Kontrolle auf 1 gesetzt, wenn die Anzahl Bits auf 1 ungerade sind. Wiederum auf 0, wenn die Anzahl gerade ist.
Zur Kontrolle wird festgestellt, ob pro Zeile eine gerade Anzahl an Bits mit dem Wert 1 vorliegt. Ist dies nicht der Fall also eine ungerade Anzahl von Bits mit dem Wert 1, so liegt in dieser Zeile einen Fehler vor.

									Partiätsbit Spalte	Fehler?
Byte 1	1	1	0	0	0	1	1	1	1	X
Byte 2	0	0	0	1	1	1	0	1	0	X
Byte 3	1	1	1	1	1	1	1	0	1	X
Byte 4	0	0	0	0	0	1	1	1	1	X
Byte 5	0	1	1	0	0	1	1	0	0	X
Byte 6	1	0	0	1	0	0	1	0	1	X
Byte 7	1	0	1	0	1	0	0	0	0	Ja
Byte 8	0	1	0	1	0	1	1	0	0	X
Partiätsbit Spalte	0	0	1	0	1	0	1	1		
Fehler	X	X	X	X	X	X	Ja	X		

Im Beispiel liegt somit ein Fehler im 7. Byte und in der 7. Bitspalte. Somit im 7. Byte im 7. Bit.

Hamming Code

Der Hemming Code ist eine weitere Möglichkeit Fehler zuerkennen. Wobei beim Hemming Code die originale Nachricht in einen Rahmen eingepflegt wird, wo das jeweilige 2er-Potenzbit als Kontrollbit reserviert wird. Also Bit 1,2,4,8,16,…
Es werden alle Positionsnummern mit einer Eins XOR-Verknüpft.

Beispiel

Original Nachricht: 1001 0000

Platz	12	11	10	9	8	7	6	5	4	3	2	1
Data	1	0	0	1		0	0	0		0		

Position	**Wert**
12	1100 (12)
9	1001 (9)
XOR-Ergebnis	0101

Das Ergebnis wird in die Paritätsbits eingesetzt

Platz	12	11	10	9	8	7	6	5	4	3	2	1
Data	1	0	0	1	0	0	0	0	1	0	0	1

Nach der Übertragung werden die Positionswerte sämtlicher Bits (auch die der Paritätsbits) Binär dargestellt.

Position	**Wert**
12	1100
9	1001
4	0100
1	0001
XOR-Ergebnis	0000

Ergibt das Ergebnis Binär 0, so ist bei der Übertragung nichts schief gegangen.
Ist ein Fehler entstanden, so gibt das XOR-Ergebnis automatisch die Position des Fehlers aus. Durch Umkehren dieser genannten Position, wird der Fehler korrigiert.

Platz	12	11	10	9	8	7	6	5	4	3	2	1
Data	1	0	0	1	0	0	1	0	1	0	0	1

Durch Fehlübermittlung wurde das Bit 6 fehlerhaft empfangen.

Position	**Wert**
12	1100
9	1001
6	**0110**
4	0100
1	0001
XOR-Ergebnis	0110 -> Ergebnis Fehlerhaftes Bit

Um die Redundanz zu berechnen, wird die Totalanzahl Bits (T) minus die Anzahl «Original»-Bits (O) dividiert durch die Totalanzahl Bits (T).gerechnet:

$\frac{2^T - 2^O}{2^T}$ $\frac{2^{12} - 2^8}{2^{12}}$ = 0.9375 = 93.75%

Kompression

Ziele einer Kompression ist es weniger Speicherplatz zu verbrauchen, gleichzeitig jedoch auch die Übertragungszeiten zu optimieren.

Es gibt verlustbehaftete Kompression und verlustfreie Kompression. Bei der verlustbehafteten Kompression werden irrelevante Informationen weggelassen, bei der verlustfreien Kompression hingegen werden die Redundanzen ausgenutzt, und keine Informationen «gelöscht».

Bei der verlustfreien Kompression muss die «Anleitung» bei Komprimierung sowie Dekomprimierung bekannt sein.

Kompressionsrate

Die Kompressionsrate gibt an wie viel Prozent, dass eingespart werden gegenüber der nicht komprimierten Übertragung.

Sie wird folgendermassen berechnet:

$$KR = \frac{Anz\ Bits\ komprimiert}{Anz\ Bits\ unkomprimiert} = 0.XXX = XX.X\%$$

Beispiel:

Payload	Anzahl Bits (ASCII)	Kompressionsrate
Zweihundertsiebzehn Milliarden	210 Bit	0%
217’000’000’000	105 Bit	50%
2.17e11	49 Bit	23.3%

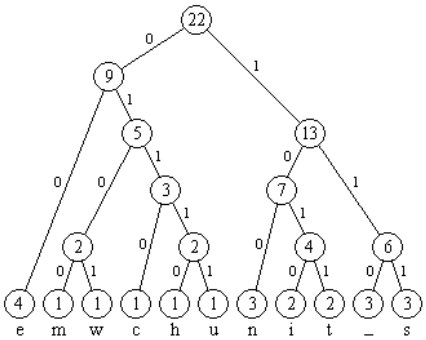
Huffman – Codierung

Die Huffman – Codierung wird ein Code generiert, welcher für den vorliegenden Text optimal gestaltet ist.

- Häufige Zeichen -> kürzester Code
- Seltene Zeichen -> längerer Code
- Es wird ein eindeutiger Bitstrom ohne Trennzeichen generiert
- Es gibt nicht DEN optimalen Code, sondern meist mehrere

Vorgehen Huffman – Codierung

1. Jedes vorkommende Zeichen wird auf ein Blatt notiert
2. Die Häufigkeit wird über das jeweilige Zeichen notiert
3. Es werden nun immer die beiden tiefsten (freien) Häufigkeitswerte (oder Knoten) nach oben zu einem Summenknoten.
4. Dies wird wiederholt, bis nur noch ein Knoten übrig ist.
5. Unter jeden Knoten wird links eine null und rechts eine eins geschrieben.
6. Nun kann der Binärcode jedes Zeichen abgelesen werden.



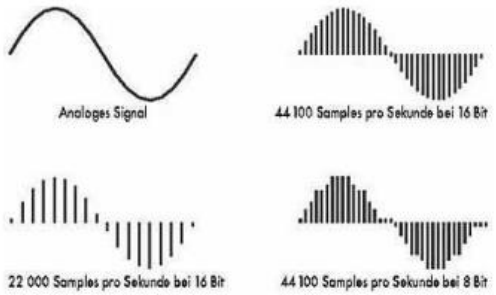
E	00
M	0100
W	0101
C	0110
H	01110
U	01111
N	100
I	1010
T	1011
[ ]	110
S	111

Einsatzzweck

Die Huffman Codierung wird in folgenden weitverbreiteten Formaten vertreten:

- JPEG
- ZIP (Implode)
- MPEG
- …

Verlustbehaftete Kompression



Verschlüsselung

Verschlüsselung ist dazu da, Daten sicher zu übertragen und zu speichern.
Sicher beruht auf folgenden drei Punkten:

- Vertraulichkeit
  - Inhalt können von unbefugten nicht eingesehen werden
- Integrität
  - Die Inhalte sind unverfälscht und vollständig
- Authentizität
  - Sender und Empfänger einander klar bekannt sein

Symmetrische Verschlüsselung

+ Schnell

- Viele Schlüssel nötig
- Ein Austausch des Schlüssels ist zuvor nötig (Preshared Key)

Asymmetrische Verschlüsselung

+ Sicherer
+ Schlüssel müssen nicht zuvor übertragen werden
- Langsamer

Hybride Verschlüsselung

Bei der Hybriden Verschlüsselung werden die Vorteile der symmetrischen und der asymmetrischen Verschlüsselung kombiniert. Zum Aushandeln des Preshared Keys, wird eine asymmetrische Verschlüsselung verwendet, die Nachricht wird anschliessend nach einem symmetrischen Verschlüsselungsverfahren übertragen.

Signatur

Um die Integrität einer Übertragung sicherzustellen, wird mittels des private Keys die Nachricht signiert. Die Prüfung der Integrität wird wiederum mit dem public Key sichergestellt.

Steganografie

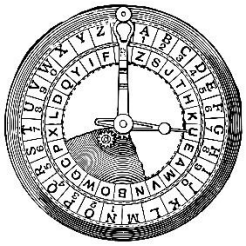
Bei der Steganografie wird die Nachricht nicht primär verschlüsselt sondern sie wird in einer anderen Datei versteckt.

- Text in Text
- Bilder in Bildern
- First Letter Messages
- Bitströme in Dateien

Monoalphabetische Verschlüsselung

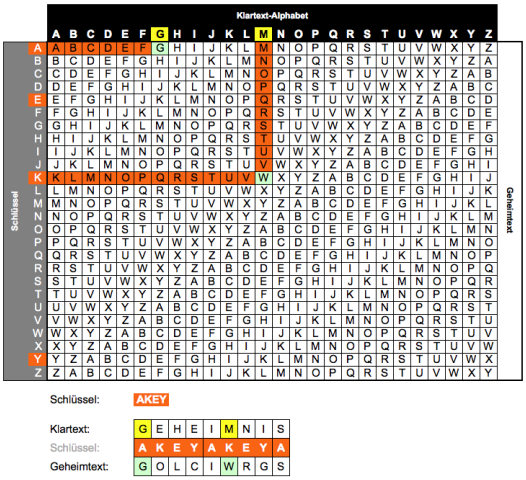
In der monoalphabetischen Verschlüsselung werden die Buchstaben des Alphabets um mehrere Buchstaben verschoben. Es entsteht ein nicht auf den ersten Blick lesbarer Text.

Die Caesar Cipher ist eine monoalphabetische Verschlüsselung welche bereits zu Zeiten von Caesar Verschlüsselung bot.



Polyalphabetische Verschlüsselung

In der polyalphabetischen Verschlüsselung werden die Buchstaben des Alphabeten genau wie in der monoalphabetischen Verschlüsselung verschlüsselt. Nur wird jeder Buchstabe mit einem anderen Caesar-Schlüssel chiffriert.



Beide Verfahren sind sehr einfach und können durch heutige Computer in Sekundenbruchteilen geknackt werden.

Moderne Chiffrierung

XOR-Chiffre

Bei der XOR-Chiffrierung wird der Plaintext XOR-Verknüpft mit dem Schlüssel.

Hash- und Einwegfunktionen

Hash- und Einwegfunktionen lassen einen Payload Chiffrieren ohne, dass dieser auf irgendwelche Weise wird zurückberechnet, werden kann.
Ein Hash kann auf keine Art und Weise zurückgerechnet werden und gilt als mathematisch «perfekt». Jedoch kann ein Hash als nicht mehr integer eingestuft werden, wenn eine sogenannte Collision Attack erfolgreich stattgefunden hat.

Um zu verstehen was eine collision attack ist, muss man wissen, dass die Änderung nur eines Bits in einem Dokument einen komplett anderen Hash ergibt.

Eine Collision Attack ist ein Zustand, wo ein Hash von zwei verschiedenen Dokumenten exakt derselbe ist, obwohl der Inhalt ein anderer ist.

Ein solches Beispiel findet man auf der Seite https://shattered.io/

CRC (Cyclic Redundancy Check)

CRC ist dazu da Fehler in der Übertragung festzustellen. Nicht umkehrbare mathematische Abbildungen sind eine Polynom-Division durch einen Standardisierten Wert.

Dabei wird der Rest der Division so zum Text addiert, dass kein Rest entsteht. Entsteht ein Rest, so ist die übertragene Message fehlerhaft übertragen oder manipuliert worden.

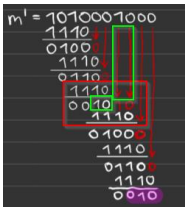
CRC Berechnung

Es wird eine Generator-Polynom *g* mit *n* Stellen benötigt.
Die Nachricht *m* wird ergänzt mit *n*-1 Nullen ergänzt.
Die Nachricht *m'* wird nun durch das Generatorpolynom «dividiert»/XOR-Verknüpft. Der entstehende Rest, ist die Prüfsumme.

Vorgehen Dividierung

1) Nachricht aufschreiben
2) XOR-Verknüpfung berechnen (unter dem Strich)
3) Nächste Stelle holen
4) Generator-Polynom mit seiner ersten 1 unter die erste 1 Schreiben
5) Wiederum XOR-Operation, usw.bis alle Stellen geholt werden

Wichtig: Nur signifikante Stellen sind zur Berechnung nehmen (häufiger Fehler):



Berechnung Prüfsumme

XOR-Operation durchführen bis alle Nullen «unten sind».

Prüfung der Prüfsumme

Kommt beim Ergebnis 0 heraus, so wurde die Übermittlung korrekt vorgenommen.

RSA-Verschlüsselung

Die RSA-Verschlüsselung ist eine asymmetrische Verschlüsselungsmethode.
Sie besteht aus zwei Primzahlen (p & q). Mittels öffentlichen Verschlüsselungsexponenten (e) wird eine Message (m) verschlüsselt. Nur mittels private Key (d) kann die Message wieder entschlüsselt werden.

Vorgehen

Verschlüsseln

p, q, e = Primzahl | m = Message | n = RSA-Zahl | c = encrypted message |

1. n = p * q
2. ø(n) = (p-1)*(q-1)
3. e * d mod(ø(n))
   3.1. d = ø(n) : e -> muss ganz Zahlig sein
4. $c = m^e \bmod(n)$

Entschlüsseln

5. $m = c^d \bmod(n)$

Beispiel

p = 17, q = 3, e = 5, m = 11

1. n = 17 * 3 = 51
2. ø(n) = (17-1)*(3-1) = 32
3. 5 * d mod(32)
   3.1. d = (32+1) : 5 -> nicht ganz Zahlig
   3.2. d = (2*32+1) : 5 = 13 -> ganz Zahlig
4. $11^5 \bmod 51 = 44 = c$
5. $m = 44^{13} \bmod 51 = 11$

Tools

- https://cryptool.org/de/cto
-