

**Allgemein:**

-Zahl1 % Zahl2 = Rest maximal so gross wie Zahl1

-Verschlüsselung Einsatzgebiet: Email, ZIP, Ebanking, Textdateien, Ordner, Tür, WLAN, Telefon, SMS

-Der für Menschen hörbare Bereich liegt bei 20Hz (Bass) bis 20000Hz (Höhen)

Speicherplatz = Samplingrate(Hz) \* Samplingtiefe(Bit) \* Anzahl Kanäle(Mono=1, Stereo=2, etc.) \* Dauer(Sec) = X Bit / 8 / 1000000 = MB

**RSA Verschlüsselung****Formeln Schlüsselerzeugung:**

für ggT gcd(x,y)

$n$  (Modulzahl) =  $p \cdot q$

$\phi(n)$  (geheime Modulzahl) =  $(p-1) \cdot (q-1)$

$e$  (teilerfremd zu  $\phi(n)$ ) = Selber definieren (am besten eine Primzahl) (Bedingung: ggT( $e$ ,  $\phi(n)$ ) = 1)

$d$  (privater Schlüssel) =  $((\phi(n) + 1) / e)$  = Ganzzahl (? versuchen bis es klappt)

**Public Key** =  $e = \dots$ ,  $n = \dots$

**Private Key** =  $d = \dots$ ,  $n = \dots$

**Formeln Verschlüsseln:**

$m$  (Nachricht) = Selber definieren (Zahl muss kleiner sein als  $n$ )

$c$  (Geheimtext) =  $m^e \bmod n$

**Formeln Entschlüsseln:**

$m$  (Klartext) =  $c^d \bmod n$  (Kann zum Teil nicht gerechnet werden, da zu grosse Zahlen)

**Beispiel Schlüsselerzeugung: Alice**

$p = 7$ ,  $q = 11$

$n = 7 \cdot 11 = 77$

$\phi(n) = 6 \cdot 10 = 60$

$e = 13$

$d = 37$  bei  $? = 8$

**Public Key** =  $e = 13$ ,  $n = 77$

**Private Key** =  $d = 37$ ,  $n = 77$

**Beispiel Verschlüsseln: Bob**

$m = 2$

$c = 2^{13} \bmod 77 = 30$

**Beispiel Entschlüsseln: Alice**

$m = 30^{37} \bmod 77 = 2$

**Kompressionsfaktor / -rate****Formeln:**

Kompressionsrate:  $uk / k$  (Rechts fix = 1) ->  $X:1$

Kompressionsfaktor:  $k / uk$  (Links fix = 1) ->  $1:X$

$k$  = Kompressionsfaktor \*  $uk$

$k$  = komprimiert,  $uk$  = unkomprimiert (original)

**Beispiel: Kompressionsfaktor = 3 / 3.77**

Dezimal = 0.796

Prozentsatz = 79.6 %

Gekürzt =  $1: (1 / (3 / 3.77)) = 1:1.257$

=>  $(1 / 1.257 = 0.796)$

=>  $(1 / 0.796 = 1.257)$

**Beispiel: Kompressionsrate = 3.77 / 3**

Dezimal = 1.257

Gekürzt =  $(3.77 / 3) : 1 = 1.257:1$

=>  $(1.257 / 1 = 1.257)$

**Wichtig:**

-Prozent deutet meistens auf Faktor hin

-Die Datei ist nur noch 79.6% gross wie die

Originaldatei = Dateigrösse: 79.6 %

-Die Dateigrösse hat um 79.6% abgenommen =

Dateigrösse: 20.4%

**Hash Funktionen****One Way Hash:**

Eine Hash Funktion berechnet aus einer beliebigen Eingabe einen Wert mit einer vorgegebenen Länge (z.B. auch ISBN, IBAN etc)

**MD5, RIPEMD128, SHA1, SHA256** unsicher, da sehr schnell errechenbar oder in Lookup Tables zu finden wenn keine weiteren Massnahmen

**Lookup Table:** Tabelle mit vorberechneten Hash Werten.

erstellen Dauer  $s = \text{Anzahl Zeichen}^{\text{Anzahl Stellen}} / \text{Anzahl parallele Berechnungen pro Sekunde}$

**Salzen** = Eine zufällige Zeichenfolge (min 20 Zeichen) welche dem Passwort angehängt wird. (Wenn möglich für jeden Benutzer ein eigenes Salz)

+ Hash kann nicht mehr gegoogelt werden

- Falls das Salz bekannt ist und die Lookup Table erstellt ist es wieder genauso unsicher wenn kein Pfeffer vorhanden ist

**Bcrypt:**

Kostenfaktor = Macht, dass der Bcrypt hash zukunftssicher ist

AnzahlRunden =  $2^{\text{Kostenfaktor}}$  (Brauchbar 8-12 Runden)

Ein richtiger Hash wird in einer Datenbank als folgender 60 Zeichen langer String gespeichert (Algorithmus:  $2y = \text{Bcrypt}$ )

\$Algorithmus\$Kostenfaktor\$SaltHashwert

\$2y\$12\$Da2S4lzH4sH

**Pfeffer** wird extern der Datenbank an einem sicheren Ort auf dem Server gespeichert und vor dem durchführen vom bcrypt hash und salz hinzugefügt

-Zum Speichern von Passwörtern, sollte eine Ein-Weg Hash-Funktion verwendet werden.

-Jedes Passwort wird mit einem eigenen Salt versehen, um den Einsatz von Rainbowtables unpraktikabel zu machen.

-Wir benötigen einen langsamen, anpassbaren Hash-Algorithmus wie Bcrypt, um Brute-Force Angriffe auszubremsen.

-Durch Zugabe eines starken Pfeffers erreichen wir, dass sich das Passwort aus dem Hashwert und dem Salt aus der Datenbank nicht mehr rekonstruieren lässt. Denn der Pfeffer fehlt. Dieser wird nicht in der Datenbank sondern an einem sicheren Ort auf dem Server gespeichert.

\$scharfesPasswort = \$passwort + \$pfeffer;

\$hash = bcrypt(\$scharfesPasswort, \$salz);

**SQL Injection:** ausführen von SQL Befehlen in Formularfeldern und somit sensible Daten von der Datenbank ausgeben.

## .zip und .tar.gz:

tar: Zusammenfassen mehrerer Dateien und Verzeichnisse in eine Datei (Archiv)  
gz: Reduzieren des Speicherplatzes durch Kompression

## Beispiele:

### Ordner in neue Datei komprimieren:

tar -czvf M114-Bildcodierung.tar.gz AB114-05

### Weiteren Ordner hinzufügen ins gleiche Archiv

**Achtung!** tar -czvf M114-Bildcodierung.tar.gz AB114-05 Zweite

### Inhalt vom Archiv anzeigen

tar -t -f M114-Bildcodierung.tar.gz

### Archiv in eine testordner wiederherstellen

tar -xvcf M114-Bildcodierung.tar.gz -C ../testordner

### Komprimierte Datei Entpacken

gunzip Dateiname.gz

.zip ist nicht immer kleiner bei kleinen Dateien (wenig Text)

Optionen von tar	
Option	Beschreibung
-h, --help	Zeigt eine vollständige Übersicht über alle Optionen.
--version	Gibt die installierte Version von tar aus.
-c	Ein neues Archiv erstellen.
-d	Daten in Archiv und im Dateisystem miteinander vergleichen.
-f	Archiv in angegebene Datei schreiben, f Daten aus angegebener Datei lesen.
-j	Archiv zusätzlich mit bzip2 (bz) komprimieren.
-J	Archiv zusätzlich mit xz (xz) komprimieren.
-k	Das Überschreiben existierender Dateien beim Erstellen aus einem Archiv verhindern.
-r	Zugriffrechte beim Erstellen erhalten.
-R	Daten in ein bestehendes Archiv anhängen.
-t	Inhalt eines Archivs anzeigen.
-u	Nur Dateien anhängen, die jünger sind als ihre Archiv-Version.
-v	Ausführliche Ausgabe aktivieren. Hierbei ist zu beachten, dass man das möglichst am Anfang des Befehls anhängt, wenn mehrere Optionen kombiniert werden, z.B. würde -rv zu einer Fehlermeldung führen. Korrekt wäre -rvf.
-w	Jede Aktion bestätigen.
-x	Daten aus einem Archiv extrahieren.
-z	Archiv zusätzlich mit gzip (gz) komprimieren.
-Z	Archiv zusätzlich mit compress (c) komprimieren.
-A	Inhalt eines bestehenden Archivs in ein anderes Archiv kopieren.
-B	Mehrfachtes Archiv anhängen/anzeigen/extrahieren.
-L	Medium wechseln, wenn ZAPF-Köcher geschrieben sind.
-P	Archiv nach dem Schreiben prüfen.

## Beispiele von der Tar Seite

### Einem Archiv eine Datei hinzufügen

tar -rf archiv.tar datei\_1.txt

### Ein Archiv mit dem Namen archiv.tar mit den Dateien datei\_1.txt und allen Dateien vom Typ \*.pdf anlegen

tar -cf archiv.tar datei\_1.txt \*.pdf

### Ein Archiv mit dem Ordner daten inklusive aller Unterordner und Dateien anlegen

tar -cf archiv.tar daten/

### Ein Archiv anlegen, zwei Dateien hinzufügen und nachträglich mit gzip komprimieren

tar -czf archiv.tar.gz datei\_1.txt datei\_2.txt

### Den Inhalt eines (komprimierten) Archivs ausführlich anzeigen

tar -tvf archiv.tar

### Fügt nur Dateien hinzu, wenn sie neueren Datums sind als ihr Gegenstück im Archiv / Bei Aktualisierung keine Unterordner berücksichtigt

tar -uf archiv.tar daten

### Den Inhalt eines Archivs mit dem Dateisystem vergleichen

tar -dvf archiv.tar

### Alle Dateien aus einem Archiv im aktuellen Ordner extrahieren

tar -xf archiv.tar

### Alle Dateien aus einem mit gzip komprimierten Archiv im ursprünglichen Ordner extrahieren

tar -xzf archiv.tar.gz -C /

### Alle Dateien in ein bestimmtes Verzeichnis extrahieren (das Ziel-Verzeichnis muss bereits existieren)

tar -xzf archiv.tar.gz -C /PFAD/ZUM/ORDNER

### Eine bestimmte Datei aus einem Archiv extrahieren / Der Pfad / Dateiname muss im Archiv genau so existieren

tar -xzf archiv.tar.gz Pfad/Dateiname

## Huffman Code:

1. Buchstaben zählen nach Häufigkeit ( \_ nicht vergessen) (Total zusammenzählen)

2. Baum bilden (1 Möglich, 2 Möglich, 3 ....) (links 0, rechts 1)

3. Tabelle ableiten A = 00....

4. Bei allen Buchstaben hinschreiben wie viele bits und mit Anz. Multiplizieren, danach addieren = Neu X bits (vorher ASCII Anz Zeichen \* 8)

5. Text (Nutzdaten) Binär schreiben 00 1010 etc.

A L etc.

6	3	3	4	2	2	1	1	1	1	1	1	1
A	N	E	D	S	H	U	R	B	P	L	T	G
7	3	3	3	4	4	5	5	5	5	5	5	5
12	9	9	12	8	8	5	5	5	5	5	5	5
= 88												

D	1010
A	00
S	1011
	010
H	11000
...	

## Regeln:

-Immer tiefe Knoten machen

-Tieferliegende Knoten haben Vorrang

- Bei Punkt 4 sollten die bits aufsteigend sein

- Für die Entschlüsselung benötigt man den Baum und den Text und geht immer z.B. 00 bis zu A und dann beginnt man wieder von oben usw.

## Cäsar

**Cäsar X:** Cäsar 1 = a->b Cäsar 5 a->f

**Angriffe:** 1) Alle 26 Möglichkeiten durchtesten 2) Häufigkeitsanalyse (Buchstabe, welcher am meisten vorkommt = E und dann X für die anderen)

**Homophone Chiffre:** Anzahl Schlüssel anhand der Häufigkeit definieren A 7% = 7 Schlüssel E = 17% = 17 Schlüssel

**Angriffe auf Hom...:** 1) Anzahl Möglichkeiten riesig, das heisst es dauert sehr lange 2) nicht mehr anwendbar

## Vigenere

Klartext: A B B A F A N In Tabelle links

Schlüssel: S O U L S O U In Tabelle oben

Geheimtext: S P V L X O H In Tabelle dort wo sich Klartext und Schlüssel kreuzen

Um zurück zu verwandeln in Tabelle schauen wo der Schlüssel und X den Geheimbuchstaben gegeben haben und X ist dann der Klartext.

## Angriffsarten:

**Ciphertext only Attack:** Mallory kann den Klartext nicht. Das ist die schwierigste Art des Angriffs

**Known Plaintext Attack:** Mallory kennt den Klartext und den Geheimtext, so kann er versuchen den Schlüssel herauszufinden um künftige Nachrichten zu entschlüsseln.

**Chosen Plaintext Attack:** Mallory hat Zugriff auf den Verschlüsselungsvorgang. Er kann selber Klartexte in Geheimtexte umwandeln und möchte so den Schlüssel herausfinden.