Leibniz-Rechenzentrum
der Bayerischen Akademie der Wissenschaften

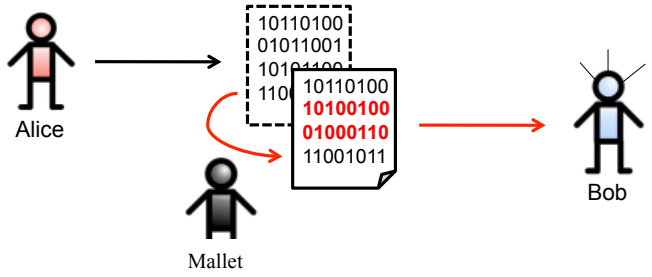
Kapitel 9:
Kryptographische Hash-Funktionen

Inhalt

- Definition: Kryptographische Hash-Verfahren
- Angriffe gegen One-Way-Hash-Funktionen
- Konstruktion von Hash-Funktionen
- Algorithmen:
  - MD5
  - SHA-3 (Keccak)

Hash-Funktionen zur Integritätssicherung

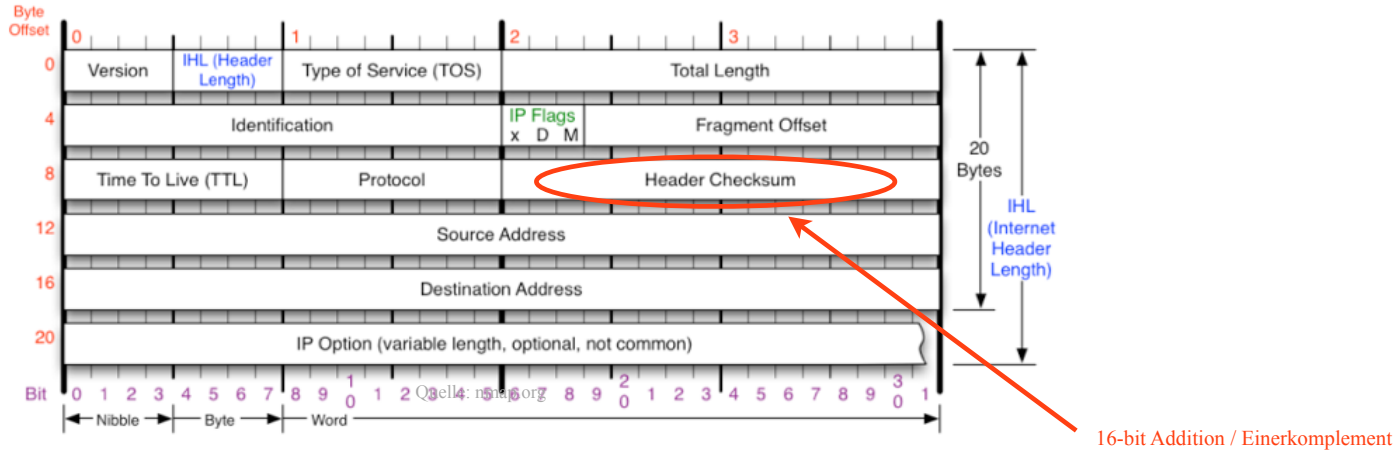
- Ziel: Sicherstellen, dass Manipulationen an einer übertragenen Nachricht erkannt werden.



- Beispiel Software-Distribution:

Herkömmliche vs. kryptographische Hash-Funktionen

- Prüfsummen dienen der Erkennung von (unbeabsichtigten) Übertragungsfehlern, z.B. beim IPv4-Header:



Version

Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.

Header Length

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

Protocol

IP Protocol ID. Including (but not limited to):

1 ICMP	17 UDP	57 SKIP
2 IGMP	47 GRE	88 EIGRP
6 TCP	50 ESP	89 OSPF
9 IGRP	51 AH	115 L2TP

Total Length

Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.

Fragment Offset

Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.

Header Checksum

Checksum of entire IP header

IP Flags

x D M

x 0x80 reserved (evil bit)
D 0x40 Do Not Fragment
M 0x20 More Fragments follow

RFC 791

Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.

- Kryptographische Prüfsummen sollen auch absichtliche Manipulationen erschweren

Grundlagen

- Hash-Funktionen
  - bilden „Universum“ auf endlichen Bildbereich ab
  - sind **nicht** injektiv
  - Bildbereich i.d.R. sehr viel kleiner als Universum
  - Kollisionen möglich:

$$\exists x, y \in U : x \neq y \quad \wedge \quad h(x) = h(y)$$

- Kryptographische Hash-Funktion H:
  - Eingabe: beliebig langes Wort m aus dem Universum U
  - Ausgabe: Hashwert H(m) mit fester Länge
  - H soll möglichst kollisionsresistent sein

Beispiel

- MD5-Hashwerte sind immer 128 Bits lang
  - egal, wie lange die Eingabe ist

```
829c11ba6dcdfe045dd1e5a77b34c05e  OOo-SDK_3.2.1_Linux_x86-64_install-deb_en-US.tar.gz
0f8abee370438e49e7ea0c2287589760  OOo-SDK_3.2.1_Linux_x86-64_install-rpm_en-US.tar.gz
35e8406c95c58b0087b9ad964faa13b8  OOo-SDK_3.2.1_Linux_x86_install-deb_en-US.tar.gz
ecc8271619ad788203cc61c7d9930522  OOo-SDK_3.2.1_Linux_x86_install-rpm_en-US.tar.gz
dddab486fd466bb1fc1a126d75919a3f  OOo-SDK_3.2.1_MacOS_x86_install_en-US.dmg
```

- Weil es nur 2^{128} verschiedene MD5-Hashwerte gibt, existieren beliebig viele Dateien mit demselben MD5-Hashwert
  - = Kollision

- Zwei sehr ähnliche, aber nicht identische Eingaben sollen nicht denselben MD5-Hashwert haben
  - = Kollisionsresistenz

- Angreifer versucht, die Nachricht m „sinnvoll“ in m‘ abzuändern, so dass md5(m) = md5(m‘)

Def. Kryptographische Hashfunktion

- Schwache Hash-Funktion H:
  - H besitzt die Eigenschaften einer Einwegfunktion
  - Hashwert H(m) = h mit |h|=k (z.B. k = 128 Bits) ist bei gegebener Nachricht m einfach zu berechnen
  - Bei gegebenem h = H(m) für $m \in A_1^*$ ist es praktisch unmöglich, eine (sinnvolle) m‘ zu finden mit:

$$m' \neq m,\ m' \in A_1^* \ \wedge\ H(m') = h$$

- Starke Hash-Funktion H:
  - H hat alle Eigenschaften einer schwachen Hash-Funktion
  - Es ist zusätzlich praktisch unmöglich, eine Kollision zu finden, d.h. ein Paar verschiedene Eingabewerte m und m‘ mit:

$$m' \neq m,\ m, m' \in A_1^* \ \wedge\ H(m) = H(m')$$

Birthday Attack auf One-Way-Hash-Funktionen

- Wie viele Personen brauchen Sie, damit mit Wahrscheinlichkeit P > 0,5 eine weitere Person mit Ihnen Geburtstag hat?
  - Antwort: 253 $P = 1 - \left(1 - \frac{1}{365}\right)^n$ (ab n=253 ist P > 0,5)
- Wie viele Personen brauchen Sie, damit mit Wahrscheinlichkeit P > 0,5 zwei Personen am selben Tag Geburtstag haben?
  - Antwort: 23 $P = 1 - \frac{365 \cdot 364 \cdots (365 - (n-1))}{365^n}$ (ab n=23 ist P > 0,5)
- Wie können Sie dieses Wissen für Angriffe gegen Hash-Funktionen nutzen?

Eine Kollision zu finden ist deutlich einfacher als
zu einem gegebenen Hash-Wert einen passenden Text!

Vorgehensweise

1. Alice sichert mit einem k Bits langen Hash eine Nachricht M
2. Mallet erzeugt $2^{k/2}$ Variationen der Nachricht M

- Die Wahrscheinlichkeit für eine Kollision ist größer 0,5.

- Wie können $2^{k/2}$ Variationen erzeugt werden?
  - Z.B. Einfügen von „Space – Backspace – Space“ Zeichen zwischen Wörtern
  - Wörter durch Synonyme ersetzen
  - .....

Beispiel für einen Brief mit 2^37 Variationen

- [Stal 98]

Dear Anthony,

{This letter is / I am writing} to introduce {you to / to you} {Mr. / --} Alfred {P. / --}

Barton, the {new / newly appointed} {chief / senior} jewellery buyer for {our / the}

Northern {European / Europe} {area / division}. He {will take / has taken} over {the / --}

responsibility for {all / the whole of} our interests in {watches and jewellery / jewellery and watches}

in the {area / region}. Please {afford / give} him {every / all the} help he {may need / needs}

to {seek out / find} the most {modern / up to date} lines for the {top / high} end of the

market. He is {empowered / authorized} to receive on our behalf {samples / specimens} of the

{latest / newest} {watch and jewellery / jewellery and watch} products, {up / subject} to a {limit / maximum}

of ten thousand dollars. He will {carry / hold} a signed copy of this {letter / document}

as proof of identity. An order with his signature, which is {appended / attached}

{authorizes / allows} you to charge the cost to this company at the {above / head office}

address. We {fully / --} expect that our {level / volume} of orders will increase in

the {following / next} year and {trust / hope} that the new appointment will {be / prove}

{advantageous / an advantage} to both our companies.

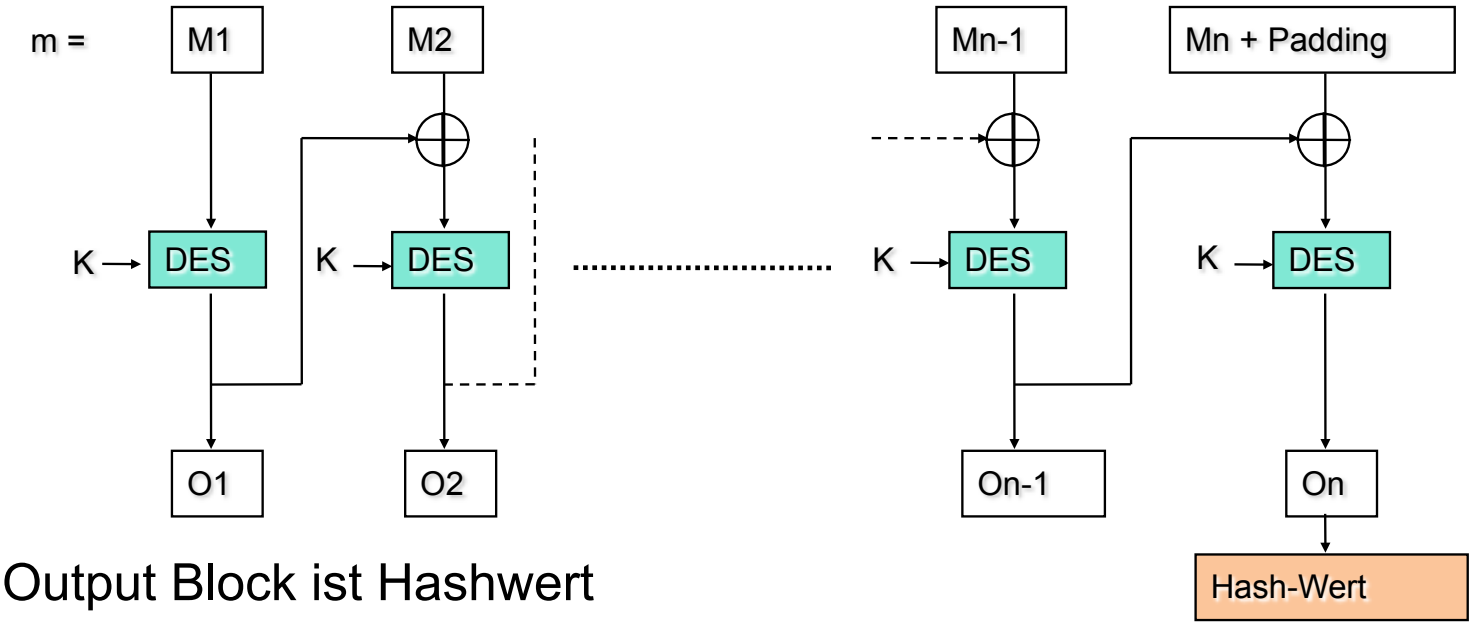
Konstruktion kryptographischer Hash-Funktionen

- Folge von Kompressionsfunktionen G
- Nachricht m wird in Blöcke Mi mit fester Länge y zerlegt
- Hash-Verfahren wird mit Initialisierungswert IV vorbelegt

- Letzter Block Mn muss ggf. auf vorgegebene Länge y „aufgefüllt“ werden (Padding)
- Als Kompressionsfunktion G können verwendet werden:
  - Hash-Funktionen auf der Basis symmetrischer Blockchiffren
  - Dedizierte Hash-Funktionen

DES als Kompressionsfunktion

- DES im Cipher Block Chaining (CBC) Mode



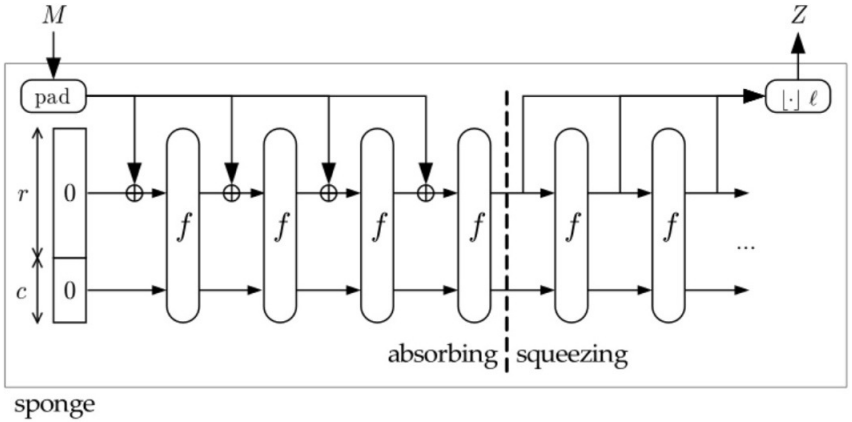
- Letzter Output Block ist Hashwert
- Länge des Hashwerts? 64 Bits

SHA-3

- 10/2012 vom NIST als Nachfolger von SHA-2 standardisiert

- 2007: Wettbewerb ähnlich zu AES-Standardisierung:
  - motiviert durch erfolgreiche Angriffe auf MD5 und SHA-1
  - 64 Einreichungen, 14 Algorithmen in engerer Auswahl, 5 Finalisten
  - Gewinner: Keccak von Bertoni, Daemen, Peeters und van Assche

- Innovativer Ansatz: Sponge-Funktion



Zwei Phasen: absorbing/squeezing

Variable Output-Länge

Bildquelle: http://sponge.noekeon.org

Keccak: Parametrisierung und Keccak-f

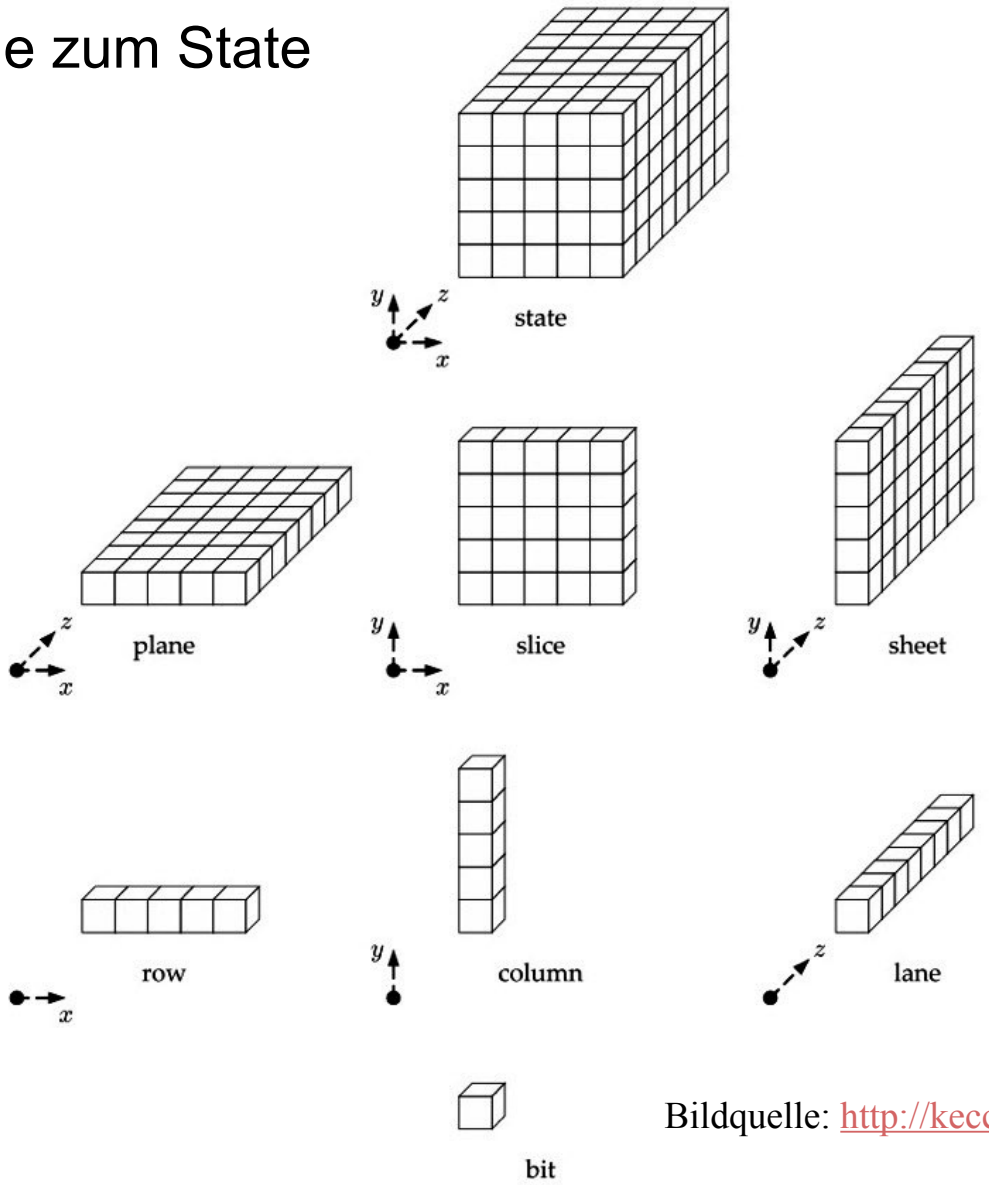
- Als SHA-3 standardisierte Varianten umfassen u.a.
  - SHA3-256: r=1152, c=448, Ausgabe abgeschnitten nach 256 Bits
  - SHA3-512: r=576, c=1024, Ausgabe abgeschnitten nach 512 Bits
- f[b] Keccak Permutationsfunktion; Breite der Perumutation
  $b = c + r = 25*2^l$
- Funktion f betrachtet State als dreidimensionales Array von GF[2]
  $a[5][5][w]$ mit $w = 2^l$, $b = c + r = 25*2^l$

  Beispiel SHA3-256: b = 1152 + 448 = 1600,

  d.h. l = 6, w = 64

- Jede Anwendung von f besteht aus nr Runden:
  nr = 12 + 2*l, d.h. für SHA3-256: nr = 24

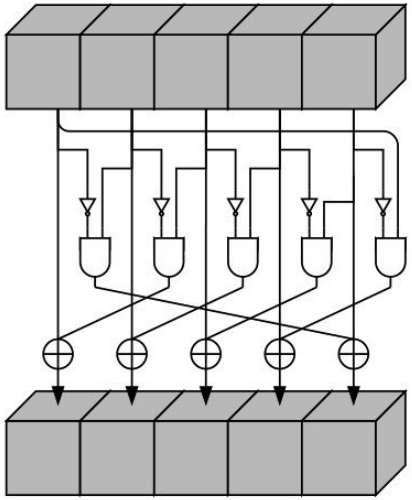
Keccak: Terminologie zum State



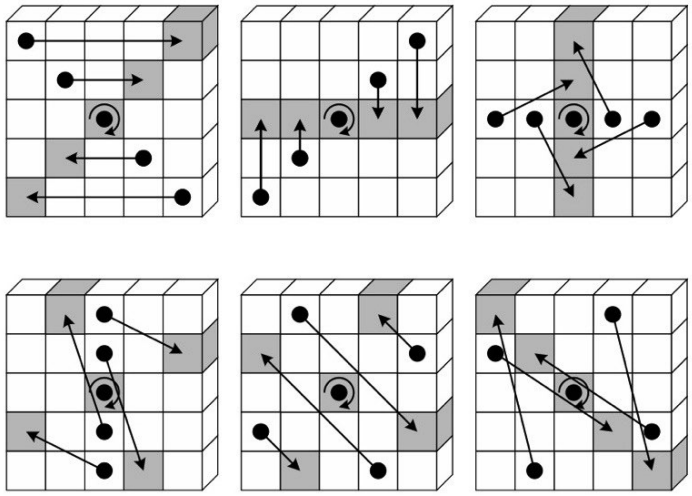
Bildquelle: http://keccak.noekeon.org/Keccak-reference-3.0.pdf

Keccak-f: Runden

- Jede Runde besteht aus fünf Schritten:
  - $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$
    - Addition von Rundenkonstanten
    - Nichtlinearität
    - Erhöhung der Diffusion in allen drei Dimensionen



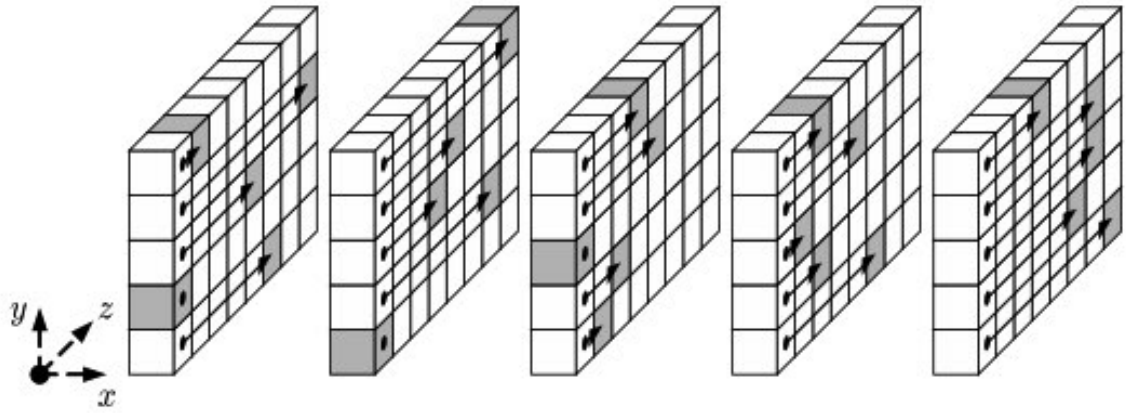
χ



π

Keccak-f: Runden

- Jede Runde besteht aus fünf Schritten:
  - $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$
    - Addition von Rundenkonstanten
    - Nichtlinearität
    - Erhöhung der Diffusion in allen drei Dimensionen



ρ

Keccak: Bewertung

- Innovativer Ansatz:
  - Vermeidet Probleme klassischer Merkle-Damgard-Konstrukte wie MD5;
  - ist entsprechend aber noch weniger von Kryptanalytikern untersucht.
  - Komplementär zu SHA-2 verwendbar.

- Variable Output-Länge
  - ermöglicht flexible Anpassung an jeweiligen Bedarf
  - Gute Eignung als PRNG für Stream Ciphers

- Effiziente Implementierung in Hard- und Software möglich

- Konservative Sicherheitsreserve durch große Rundenzahl

Praktisches Anwendungsbeispiel: Passwort-Hashes

- Krypto-Hashes werden verwendet um Passwörter (PW) zu speichern
- Bei PW-Eingabe wird Hash berechnet und mit gespeichertem verglichen
  - Hash als Einwegfunktion - Rückrechnung von Hash auf Passwort „schwer“
  - ABER: gleiches Passwort liefert gleichen Hash
  - Damit Wörterbuchangriff oder Rainbow-Tables (vgl. Kap. 12) möglich
  - Offline Angriff auf gestohlene Hash-Listen
- Abhilfe:
  - Salt: Zufallszeichenkette der beim Hash mitberechnet und mitgespeichert wird (vgl. Kap 3) - allerdings länger als beim ursprünglichen crypt - mindestens so lang wie Hash
  - Pepper: geheime Information, die nicht mit gespeichert wird:
    - gespeichert wird Salt | Hash(Passwort, Salt, Pepper)
  - Verwendung spezieller Hash-Funktionen
    - vgl. Password Hashing Competition - Gewinner Aragon
    - Speicherabhängige Hashes - damit fällt GPU-Vorteil weg