

# Hashfunktionen

Grundprinzipien  
Moderner Kryptologie

# Hashfunktion

**Input**

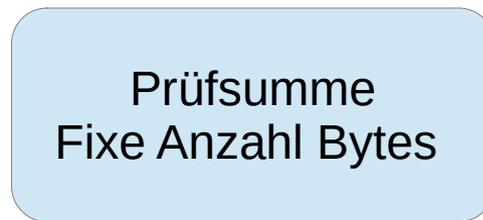


**Hashfunktion**



**Hashwert**

Prüfsumme  
Fixe Anzahl Bytes



password

SHA224



d63dc919e201d7bc4c825630d2cf25fdc93d4b2f0d46706d29038d01

# Zwingende Eigenschaften

- Gleicher Input führt immer zu gleichem Hashwert
- Jeder mögliche Input ist zulässig (beliebiger Länge)

# Gewünschte Eigenschaften

- Auch nur ein Bit Unterschied im Input führt zu “komplett” anderem Hashwert (im Sinne von zufällig anders).
- Ein Zurückrechnen ist nicht praktikabel, d.h. zu einem gegebenen Hashwert kann kein Input berechnet werden.
- Zu einem gegebenen Input einen zweiten mit gleichem Hashwert zu berechnen ist nicht praktikabel.
- Jeder Hashwert ist theoretisch möglich.
- Hashwerte können effizient berechnet werden.

# Herausforderungen

- Viele dieser Eigenschaften sind nicht beweisbar.
  - Möglich, dass jemand eine «Abkürzung» zum Zurückrechnen findet.
- Hashfunktionen können “veralten”:
  - MD5 (schon lange), SHA1 (seit einigen Jahren)

# Anwendungen

- Passwortüberprüfung
- Datenintegrität
- Corona-Tracing-App
- Digitale Unterschriften
- Block-Chains (z.B. Bitcoin)

# Datenintegrität

- Datei plus Hash davon
- Hash berechnen, vergleichen.
  - Wenn gleich, dann ok.
- Feststellen von Übertragungsfehlern
  - Schwacher Hash reicht (kein böswilliger Gegner)
- Feststellen von Manipulation
  - Starker Hash ist nötig (Sicherheit)

# Passwortüberprüfung

- Passwort speichern: Ganz schlecht!
- Hash vom Passwort speichern: weniger schlecht.
  - Problem: Hashes von häufigsten Passwörtern sind schnell erzeugt.
  - Rainbow Tables (raffinierte Tabellen um Hashes zu knacken)

# Bessere Methoden

- Zufälliges «Salz» speichern und  $\text{HASH}(\text{Salz}+\text{Passwort})$  in die Datenbank
  - Tabellen nicht mehr brauchbar, Wörterbuchattacken schon
- Hashfunktion wiederholen, d.h.  
 $\text{HASH}(\text{HASH}(\text{HASH}(\dots(\text{Salz}+\text{Passwort})\dots)))$   
so lange, bis z.B. 0.01s Rechenzeit verbraucht ist.  
Wörterbuchattacken sind so nicht mehr praktikabel

# Corona-Tracing

- Jeder generiert eine zufällige ID (kann täglich wechseln)
- Geräte stellen Kontakt fest (Distanz, Zeit)
- Einigen sich auf Zeitstempel  $z$  (z.B. 2021-06-02T10:16:46)
- Beide berechnen  $h = \text{HASH}(z+ID)$  und senden Resultat
- Beide speichern  $z$  und das  $h$  vom anderen Gerät.
- ID  $p$  einer positiven Person wird publiziert
- Gerät berechnet alle  $\text{HASH}(z+p)$  und vergleicht mit  $h$

# Einfache Prüfsummen

- Meist die letzte Ziffer von wichtigen Nummern wie
  - Kreditkarte, Kontonummer, Einzahlungsscheinreferenznummer
  - Strichcodes (z.B. EAN)
  - ISBN
- Feststellen einfacher Tipp- oder Lesefehler

# Ablauf Lektion

- Einfache Prüfsummen
  - Von Hand
  - Evtl. mit Python
- Simulation der Corona-Tracing-App
- Berechnung von (seriösen) Hash-Werten in Python
- Evtl. Block-Chain Teil 1