

Hashing

IT-Technik Netzwerkgrundlagen

Sebastian Meisel

13. Februar 2023

1 Was ist Hashing

Hashing ist eine Technik, bei der große oder komplexe Daten in eine kürzere, einzigartige Darstellung umgewandelt werden. Diese kürzere Darstellung, auch bekannt als Hashwert, dient als Fingerabdruck für die ursprünglichen Daten und kann verwendet werden, um die Integrität dieser Daten zu überprüfen. Wenn sich auch nur ein kleiner Teil der ursprünglichen Daten ändert, ändert sich auch der Hashwert vollständig.

Hashing findet in vielen Anwendungen Verwendung, einschließlich der Überprüfung der Datensicherheit, der Erstellung digitaler Signaturen und der Verwaltung von Passwörtern.

1.1 Welche Eigenschaften machen einen guten Hashing-Algorithmus aus?

Ein guter Hashing-Algorithmus muss folgende Eigenschaften aufweisen:

- **Eindeutigkeit:** Jeder Eingabe sollte einen eindeutigen Hashwert erzeugen.
- **Kollisionsresistenz:** Es sollte sehr unwahrscheinlich sein, dass zwei unterschiedliche Eingaben denselben Hashwert erzeugen.
- **Unumkehrbarkeit:** Es sollte schwierig sein, aus einem Hashwert die ursprüngliche Eingabe zu berechnen.
- **Ausgeglichene Verteilung:** Die Hashwerte sollten gut verteilt sein, so dass keine bestimmten Werte häufiger vorkommen als andere.
- **Konstanter Zeitaufwand:** Der Zeitaufwand zur Berechnung des Hashwerts sollte unabhängig von der Größe oder dem Inhalt der Eingabe konstant sein.
- **Robustheit:** Kleine Änderungen an der Eingabe sollten zu signifikanten Änderungen des Hashwerts führen.

2 Welche (gängigen, kryptographischen) Hashalgorithmen gibt es?

Für Hashing gibt es eine Vielzahl von Anwendungsmöglichkeiten, z. B. in der Programmierung und in Datenbankanwendungen. Hier geht es aber um **kryptographische** Hashingalgorithmen, die zur Verifizierung der **Integrität** von Daten verwendet werden, z. B. bei der Verschlüsselung mit TLS oder bei SSH- und VPN-Verbindungen.

Hier spielen vor allem **MD5 (veraltet!)** und **SHA** eine Rolle.

2.1 Message-Digest Algorithm 5 (MD5)

MD5 war lange Zeit der Standardalgorithmus für kryptographisches Hashing. Allerdings gilt er inzwischen als veraltet, da es möglich ist gezielt **Kollisionen** zu erzeugen. Das bedeutet, dass es möglich ist gezielt beliebige Daten so zu manipulieren, dass sie denselben Hashwert erzeugen, wie bestimmte andere Daten.

So erzeugen die beiden folgenden Linux-Binaries denselben MD5 Hashwert:

- hello
- erase

```
md5sum {hello,erase}
```

```
| da5c61e1edc0f18337e46418e48c1290 | hello |
| da5c61e1edc0f18337e46418e48c1290 | erase |
```

3 Secure Hash Algorithm (SHA)

Hierbei handelt es sich eine **Gruppe** von **kryptographischen** Hashfunktionen, die aktiv weiterentwickelt werden. Während bereits die Möglichkeit von **Kollisionsangriffen** auf **SHA-0** und **SHA-1**, gilt **SHA-2** aktuell als sicherer Standard.

Dabei gibt es verschiedene Varianten, die sich in der **Bitlänge** des Hashwertes unterscheiden:

- SHA-224
- SHA-256
- SHA-384
- SHA-512

Dabei gilt, dass mit der Bitlänge sowohl die **Sicherheit**, als auch der **Rechaufwand** zum Erzeugen des Hashwertes steigt.

3.0.1 Nachfolger

Als Nachfolger von SHA-2 wurde 2004 nach einem Wettbewerb der **SHA-3**-Algorithmus standardisiert. Ist ist deutlich zukunftssicherer als SHA-2, aber noch nicht sehr weit verbreitet.

Auch dieser Algorithmus ist in den vier Bitlängen 224, 256, 384 und 512 verfügbar. Daneben gibt es auch noch die Varianten SHAKE-128 und SHAKE-256

4 Weitere Algorithmen

Es gibt eine Reihe weiterer Algorithmen, die jedoch nicht so verbreitet sind. Von diesen gelten folgende als sicher:

- BLAKE (2. Platz im Wettbewerb nach SHA-3).
- RIPEMD-160
- Whirlpool
- Tiger
- ShangMi 3 (SM3) - chinesischer Algorithmus.

5 Befehl zum erzeugen von Hash-Prüfsummen

- Windows - unterstützte Algorithmen sind MD5, SHA1, SHA256, SHA384, SHA512:

```
Get-FileHash -Algorithm SHA256 Datei
```

- Linux:

```
md5sum
sha1sum
sha224sum
...
sha256sum
```

- Linux mit OpenSSL

```
openssl list -digest-algorithms
```

dann z. B.:

```
openssl dgst -sha3-256 hello
```

```
: SHA3-256(hello)= 0f5aa73890600ec25491c0f1707b370bb406c533622f508319c9f50c782a3f73
```