Algorithmen und Datenstrukturen SoSe25

-Assignment 8-

Moritz Ruge
Matrikelnummer: 5600961
Lennard Wittenberg
Matrikelnummer: —

Juni 2025

Problem 1: Kryptographische Hashfunktionen und Blockchain

a) Kryptographische Hashfunktionen in Scala

Welche kryptographischen Hashfunktionen sind in Scala implementiert? Wie kann man sie verwenden?

Lösung: Non-cryptographic hashfunctions

Scala3 hat keine eigene Kryptographische Hashfunktion Implementiert (jedenfalls habe ich nichts gefunden).

In Scala hat man die Möglichkeit interne Hashfunktionen zu benutzten wie z.B. mit hashCode() methode[4]:

```
scala> val result = "hello".hashCode()
val result: Int = 99162322
```

Dies dient aber nicht der Kryptographischen Verschlüsselung von Werten, da es hierbei zu viele Kollisionen kommt, eher ist es zur Kontrolle von Werten gedacht.

Eine weitere Möglichkeit ist es über eine zusätzliche Scala Library zusätzliche Hashfunktionen zu benutzen: *scala.util.hashing.Hashing* [2]

```
import scala.util.hashing.Hashing


@main def run(): Unit =

  val h = summon[Hashing[String]]
  val hashWert = h.hash("Hallo")

  println(hashWert)
```

```
Output: 69490486
```

MurmurHash3

Oder auch eine Implementierung von MurmurHash3 von Rex Kerr. Auch dieser ist aber ein *non-cryptographic hashing algorithm* [3]

```
import scala.util.hashing.MurmurHash3


@main def run(): Unit =

  val text = "Hallo Welt"
  val hashWert = MurmurHash3.stringHash(text)

  println(hashWert)
```

```
Output: -608680269
```

Kryptographische Hashfunktionen

Um Kryptographische Hashfunktionen in Scala zu benutzen, müssen wir auf Bibliotheken von Java zurückgreifen. Um dies zu tun, Importieren wir z.B.:*java.security.MessageDigest* - für die Nutzung von SHA-256, MD5 oder auch SHA-1.[1]

```
import java.security.MessageDigest
```

Um z.B.: SHA-256 zu verwenden, müssen wir die vorgefertigten Methoden, getInstance(), digest(), benutzen

```
import java.security.MessageDigest


@main def run(): Unit =

  val message = "Hello World"
  val sha256 = MessageDigest.getInstance("SHA-256")
  val hashWert = sha256.digest(message.getBytes("UTF-8"))

  println(hashWert)
```

```
Output: [B@45820e51
```

- MessageDigest - ruft das Objekt auf
- getInstance() - Returns a MessageDigest object that implements the specified digest algorithm.
- digest - Performs a final update on the digest using the specified array of bytes, then completes the digest computation.

Da wir bei *println(hashWert)* eine Standard-toString-Ausgabe von einem Java-Array erhalten(also in Bytes), müssen wir die Ausgabe noch einmal in Hex-Zahlen umwandeln:

```
import java.security.MessageDigest


@main def run(): Unit =

  val message = "Hello World"
  val sha256 = MessageDigest.getInstance("SHA-256")
  val hashWert = sha256.digest(message.getBytes("UTF-8"))

  // Bytes nach Hex-String umwandeln
  val hashHex = hashWert.map("%02x".format(_)).mkString

  println(hashHex)
```

```
Output: a591a6d40bf420404a011733cfb7b190d62c65bf0bcda32b57b277d9ad9f146e
```

b) Verkettete Liste mit Hashreferenzen

Implementieren Sie in `Scala` eine einfach verkettete Liste mit Hashreferenzen. In den Knoten der einfach verketteten Liste sollen `String`-Objekte gespeichert werden. Verwenden Sie dazu eine kryptographische Hashfunktion wie in Teil (a).

c) Nonce und Hash mit Nullen am Ende

Fügen Sie zu den Knoten Ihrer einfach verketteten Liste jeweils ein *Nonce* hinzu, und stellen Sie sicher, dass die Hashwerte in den Referenzen alle mit acht Nullen (in der Binärdarstellung) enden.

Wie viele Versuche sind dazu im Durchschnitt nötig?

References

[1] Oracle Docs. *SHA-256: Java-Scala*. URL: https://docs.oracle.com/javase/8/docs/api/java/security/MessageDigest.html. (accessed: 19.06.2025).

[2] Scala 3 Docs. *Hashing*. URL: https://www.scala-lang.org/api/current/scala/util/hashing/Hashing$.html. (accessed: 19.06.2025).

[3] Scala 3 Docs. *MurmurHash3*. URL: https://www.scala-lang.org/api/current/scala/util/hashing/MurmurHash3$.html. (accessed: 19.06.2025).

[4] Geeksforgeeks. *Scala String hashCode() method with example*. URL: https://www.geeksforgeeks.org/scala/scala-string-hashcode-method-with-example/. (accessed: 19.06.2025).