SHA-1

Lus Spachmann

FSU Jena

21.01.2022

SHA-1

- Hashalgorithmus mit 160 Bit Ausgabe
- Vorgestellt in 1993
- Verbessert in 1995
- Zählt nicht mehr als sicher
- Kollisionspaar gefunden mit Aufwand $< 2^{63}$
- Konzeptionell ähnlich wie SHA-2

Algorithmus: Variableninitialisierung

- Alle variablen außer mL sind unsigned 32-Bit
- mL ist unsigned 64-Bit

$$h_0 = 0x67452301$$
$$h_1 = 0xEFCDAB89$$
$$h_2 = 0x98BADCFE$$
$$h_3 = 0x10325476$$
$$h_4 = 0xC3D2E1F0$$
$$mL = \text{Textlänge in Bit}$$

Algorithmus: Pre-Processing

- Hänge Bit '1' der Nachricht hinzu
- Hänge genügend '0' Bits hinzu, bis Länge kongruent zu 448(mod 512) ist
- Hänge mL an
- Falls $8|mL$, kann statt Bit '1' auch Byte $0x80$ angehängt werden
- Teile Nachricht in 512-Bit Blöcke auf

Algorithmus: Verarbeitung der Blöcke (1)

- Unterteile 512-Bit Block in 32 Bit Wörter $w_0, \ldots, w_{15}$
- Erweitere auf 80 Blöcke durch

$$w_i = (w_{i-3} \text{ xor } w_{i-8} \text{ xor } w_{i-14} \text{ xor } w_{i-16}) \text{ leftrotate } 1$$

  für $16 \leq i < 80$ (Rotation zyklisch)
- Initialisiere Variablen:

$$a = h_0$$
$$b = h_1$$
$$c = h_2$$
$$d = h_3$$
$$e = h_4$$

Algorithmus: Verarbeitung der Blöcke (2) I

1: **for all** $i \in \{0, ..., 79\}$ **do**
2: **if** $0 \leq i \leq 19$ **then**
3: $f = (b \text{ and } c) \text{ or } ((\text{not } b) \text{ and } d)$
4: $k = 0x5A827999$
5: **else if** $20 \leq i \leq 39$ **then**
6: $f = b \text{ xor } c \text{ xor } d$
7: $k = 0x6ED9EBA1$
8: **else if** $40 \leq i \leq 59$ **then**
9: $f = (b \text{ and } c) \text{ or } (b \text{ and } d) \text{ or } (c \text{ and } d)$
10: $k = 0x8F1BBCDC$
11: **else if** $60 \leq i \leq 79$ **then**
12: $f = b \text{ xor } c \text{ xor } d$
13: $k = 0xCA62C1D6$
14: **end if**

Algorithmus: Verarbeitung der Blöcke (2) II

15: $tmp = (a \text{ leftrotate } 5) + f + e + k + w_i$
16: $e = d$
17: $d = c$
18: $c = b \text{ leftrotate } 30$
19: $b = a$
20: $a = tmp$
21: **end for**
22: $h_0 = h_0 + a$
23: $h_1 = h_1 + b$
24: $h_2 = h_2 + c$
25: $h_3 = h_3 + d$
26: $h_4 = h_4 + e$

Algorithmus: Finaler Output

- $hash = h_0||h_1||h_2||h_3||h_4$
- '||' bedeutet angehängt
- Alle Umwandlungen zwischen Zahlen und Bytes in Big-Endian

Aufgabe

- Implementiert SHA-1
- Testwert:

$$\text{SHA-1}(\text{““}) = \text{da39a3ee5e6b4b0d3255bfef95601890afd80709}$$