Algorithmische Anwendungen WS 2005/2006

Jens Haag & Suna Atug

Gruppe: D Grün

Projektarbeit: Message Digest Algorithm 5

Schwachstellen der MD5-Verschlüsselung, Beispiele, Anwendungen

Message Digest Algorithm 5
Entwickelt im April 1992 von Ronald L. Rivest, RFC 1321

Inhaltsverzeichnis

1. Was versteht man unter einem Hash-Verfahren?

MD5 ist eine sehr verbreitete kryptographische Hash-Funktion, die einen 128-Bit-Hashwert erzeugt.

In einem Hash-Verfahren werden Hashwerte erzeugt. Diese sind auch unter dem Begriff Prüfsumme bekannt.
Dabei wird immer eine Nachricht variabler Grösse in eine festgesetzte 128-Bit Länge Prüfsumme gewandelt. Jedoch kann nicht aus der Prüfsumme die Nachricht rekonstruiert werden, aus diesem Grund ist es nicht direkt eine Verschlüsselung wie z. B. Blowfish. Daher sind Hashwerte mit Fingerabdrücken zu vergleichen, z.B. kann man Dokumente signieren oder Passwörter verschlüsseln. Weitere bekannte Hash-Verfahren sind unter anderem SHA, HAVAL.

Eine gute Hash-Funktion kennzeichnet sich daraus, dass es unmöglich sein muss, den Fingerabdruck vorherzusagen ohne ihn aus der Originaldatei zu berechnen. Wenn dies möglich wäre, wäre die Sicherheit der Hashfunktion nicht mehr gegeben. Beliebige Nachrichten könnten so generiert werden, dass sie den gleichchen Hashwert haben, obwohl der Inhalt der Nachricht verschieden ist. Jede auch nur kleine Änderung in einer Nachricht berechnet eine komplett neue Prüfsumme.

1.1 Beispiele zum MD5 Hash-Verfahren:

Die 128 Bit langen MD5-Digests (Hashes) werden als 32-stellige Hexadezimalzahl notiert. Beispiel (generiert mit dem unter Linux verwendeten Prgramm „md5sum") der Text „Onkel Franz ist super" mit der generierten Prüfsumme:

```
Text: „Onkel Franz ist super!"
md5summe: „f9ab17956e69d0bd5f09af8d2987ded5"
```

Jede auch nur so kleine Änderung in der Nachricht ändert die Prüfsumme.
Unser vorheriges Beispiel:

```
Text: „Onkel Franz ist super!"
md5summe: „f9ab17956e69d0bd5f09af8d2987ded5"
```

Neue Prüfsumme, nach kleiner Änderung im Text:

```
Text: „Onkel Frank ist super!"
md5summe: „d364d01fb67db0ce2d9148a3280c8f3a"
```

2. Geschichte von MD5

Der Message Digest Algorithm 5 (MD5) wurde im April 1992 von Ronald L. Rivest entwickelt und steht u. a. im RFC 1321. MD5 wird in vielen Verschlüsselungsprogrammen eingesetzt, wie z.B. in PGP. MD5 ist vor allem unter Unix/Linux und auch im Internet weit verbreitet.

3. Wo wird MD5 eingesetzt?

In vielen Bereichen der elektronischen Datenverarbeitung wird MD5 bzw. das Hash-Verfahren verwendet. Es gibt vier wesentliche Einsatzbereiche für MD5:

1. Auffinden von Dateien
2. Digitale Signatur
3. Vergleich großer Texte
4. Übertragungsfehler erkennen

Auf diese vier Bereiche wird nachfolgend näher drauf eingegangen.

3.1 Auffinden von Dateien

Um in einer Datenbank Daten zu finden kann der Hashwert benutzt werden.
Benutzer Passwörter beispielsweise werden in Content Management Systemen (CMS) oder aber auch unter Linux aus Sicherheitsgründen nicht im Klartext gespeichert sondern nur die jeweilige Prüfsumme des Passwortes.

<u>Das Anmeldeszenario folgt wie folgt ab:</u>

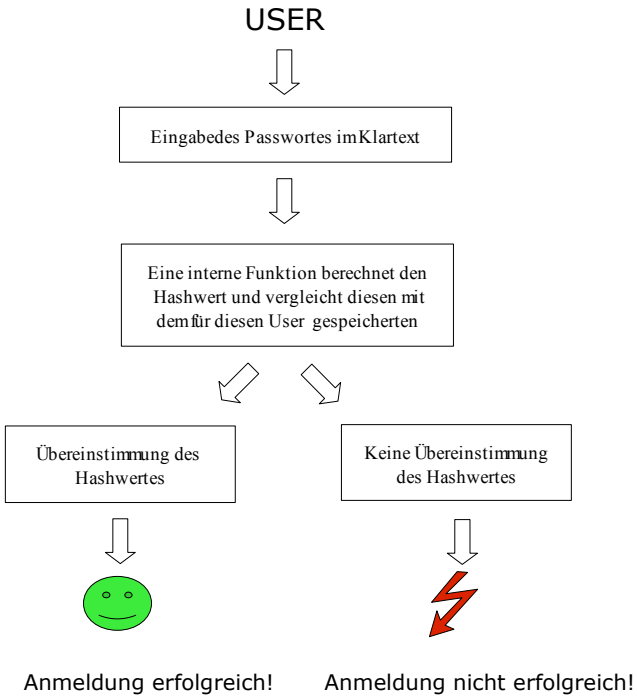

Abb. Anmeldeszenario

3.2 Digitale Signatur

Verifizierbar sind Dokumente durch digitale Signatur. Wird beispielsweise ein wichtiges Dokument per Email versendet, sollte auch der entsprechende Hashwert der Datei mit versendet werden. Der Empfänger erstellt dann auch den Hashwert der erhaltenen Datei. So kann verglichen werden, ob der mit versendete Hashwert und der selbst erstellte Hashwert identisch sind. Sind beide Prüfsummen identisch, so handelt es sich um das unveränderte Original.

3.3 Vergleich großer Texte

Vergleich von zwei ähnlich großen Dateien: Statt das die beispielsweise 30 Seiten eines Textes durchzusehen, ob auch wirklich jeder Buchstabe gleich ist, kann einfach der „kurze" Hashwert beider Dokumente verglichen werden. Dieses Verfahren nimmt weniger Zeit in Anspruch statt das buchstabenweise vergleichen.

3.4 Übertragungsfehler erkennen

Ein Autor hat z.B. auf seiner Internetseite eine Datei zum runterladen frei gegeben und den Hashwert dazu angegeben. Nach dem herunterladen der Datei kann durch die Erstellung des Hashwertes diese und die auf der Internetseite angegebenen Hashwert verglichen werden ob es einen Datenverlust gab.

4. Der Algorithmus

- Die Eingabenachricht ist eine beliebig lange, nichtnegative Integerzahl.
- Der Algorithmus ist unterteilbar in vier Schritte:

    1. Padding
    2. Länge anhängen
    3. MD-Buffer initialisieren
    4. Transformation

4.1 Padding

- In folgender Form liegt die Eingabenachricht vor: $m_0,m_1,m_2,...,m_{b-1}$, Länge = b (in Bit)
- MD5 arbeitet mit 32Bit-Worten in 16er-Blöcken (16x32Bit=512Bit)
- Es werden so lange Füllbits an die ursprüngliche Eingabenachricht angehängt, bis die Länge mod 512 448 ergibt. D.h. es bleiben 64Bit „frei" bis zum nächsten vielfachen von 512.
- Vorgehen beim Padding:
    - anhängen eines einzelnen Bits „1"
    - „0"Bits werden hinzugefügt, bis die o.g. Forderung erfüllt ist.
      (Mindestens 1 und höchstens 512Bit werden an die ursprüngliche Eingabenachricht angehängt)

4.2 Länge anhängen

- Wie oben genannt wird die Nachrichtenblöcke in 512 mod 448 Bit aufgefüllt. In die restlichen 64 Bit wird die Länge der Eingabenachricht in Form einer Integerzahl angehängt.

4.3 MD-Buffer initialisieren

- Der Hauptalgorithmus von MD5 arbeitet mit einem 128-Bit-Puffer der verwendet wird, um Zwischen- und Endergebnisse der Hash-Funktion zwischenzusspeichern. Der Speicher ist als 4 32-Bit Register angelegt, welche mit hexadezimalen Werten A, B, C, D gefüllt werden. Diese sind wie folgt initialisiert:

  (A und B sind aufwärts von 0-F sortiert C und D sind abwärts von F-0 sortiert):

  A: 01 23 45 67
  B: 89 AB CD EF
  C: FE DC BA 98
  D: 76 54 32 10

4.4 Transformation

- Auf diesen Puffer wird die Verschlüsselungsfunktion/Komprimierungsfunktion mit dem ersten 512-Bit-Block als Schlüsselparameter aufgerufen. Die Behandlung eines Nachrichtenblocks geschieht in vier einander ähnlichen Stufen, von Kryptografen „Runden" genannt. Jede Runde ist aus 16 auf einer nichtlinearen Funktion „F", modularer Addition und Linksrotation basierenden Operationen zusammen gesetzt. Es gibt vier mögliche „F"-Funktionen, in jeder Runde wird davon eine andere verwendet:

$$F(X,Y,Z)=(X\wedge Y)\vee(\neg X\wedge Z)$$

$$G(X,Y,Z)=(X\wedge Y)\vee(\neg X\wedge Z)$$

$$H(X,Y,Z)=X\ xor\ Y\ xor\ Z$$

$$I(X,Y,Z)=Y\ xor\ (X\vee\neg Z)$$

- Auf das Ergebnis wird dieselbe Funktion mit dem zweiten Nachrichtenblock als Parameter aufgerufen, und so weiter bis zum letzten 512-Bit-Block. Als Ergebnis wird wiederum ein 128-Bit-Wert geliefert, die MD5-Summe.

- Beim Durchlauf von den vier Runden spielen die folgenden Variablen r, k und i eine Rolle, diese sind wie folgt definiert:

  r = Rotationskonstante, die wie folgt definiert ist:

  Runde 1:
  r[0..15] := {7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22}

  Runde 2:
  r[16..31] := {5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20}

  Runde 3:
  r[32..47] := {4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23}

  Runde 4:
  r[48..63] := {6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21}

  k = k[i] := floor(abs(sin(i + 1)) × 2^32) ; Es wird jedoch nur der binäre Nachkommateil von Sinus beachtet!

  i = [0...63]

Ablauf:

Aktuelle Worte des Buffers werden zwischengespeichert:
A=AA
B=BB
C=CC
D=DD

Es gibt 4 Runden. Pro Runden werden 16 Operationen ausgeführt, da jedes Wort des aktuellen Schlüsselparameters bzw. Blocks in die Berechnung einfließen muss.

In jeder Runde werden die Variablen (AA, BB, CC, DD) mit den o.g. Funktionen (F, G, H, I) logisch miteinander verknüpft.

Es folgt eine Linksrotation um eine variable Anzahl Bits und eine weitere Addition von einer der logisch verknüpften Variablen.

Diese werden in den o.g. vier Variablen gespeichert.

Abschließend werden die neuen Werte auf die alten addiert.

5. Angriffsmethoden / Kollisionen

Es gibt verschiedene Typen von Angriffen, die die Qualität einer kryptografischen Hashfunktion beurteilt:

1. Preimage-Angriff
2. Kollisionsangriff
3. Brute-Force-Methode
4. Analyse-Methode

5.1 Preimage-Angriff

Wie schwer kann es sein, wenn man einen vorgegebenen Hashwert hat, daraus eine Nachricht zu erzeugen, die den selben Hashwert hat?

5.2 Kollisionsangriff

Wie schwer kann es sein, zwei verschiedene Nachrichten zu finden, die die gleiche Prüfsumme haben?

Kollisionen finden heißt, man kennt ein M (Text) und sucht ein M' (Kollision), so dass hash(M) = hash(M'). Wenn man nur den Hashwert hat, ist es weiterhin schwierig, eine Kollision für den Hash zu finden.

5.3 Brute-Force-Methode

Wird laut Wikipedia wie folgt definiert:

Brute-Force-Methode *(auf Deutsch etwa: "Methode der rohen Gewalt") ist der Fachbegriff für eine Lösungsmethode schwerer Probleme aus dem Bereich der Informatik und der Spieltheorie, die auf dem Ausprobieren aller (oder zumindest eines erheblichen Teils der in Frage kommenden) Varianten beruht.*

1994 legten Paul C. Van Oorschot und Michael J. Wiener das Konzept einer Brute-Force-Attacke auf MD5 vor, diese sollte damals mit Hilfe einer 10 Millionen US-Dollar teuren fiktiven Rechners durchgeführt werden. Dieser Rechner war speziell für diese Aufgabe ausgelegt. Theoretisch sollte damit möglich sein innerhalb von 24 Tagen eine Kollision in MD5 zu finden.

Das Projekt MD5CRK startete dann im März 2004. Dieses Projekt wollte eine Kollision finden , was 0 Dollar kosten sollte. In diesem Projekt sollte das Konzept des verteilten Rechnens genutzt werden.

5.4 Die Analyse-Methode

Im August 2004 fanden chinesische Wissenschaftler (Xiaoyun Wang, Dengguo Feng, Xuejia Lai, Honbo Yu) vom „The School of Mathematics and System Science, Shandong University, Jinan250100, China , Kollisionen für die vollständige MD5-Funktion.

Ihr erster Angriff, der eine Stunde dauerte, wurde auf einer IBM P690-Cluster durchgeführt. Davon ausgehend ließen sich weitere Kollisionen innerhalb von maximal fünf Minuten finden. Der Angriff der chinesischen Forscher basierte auf Analysen.

Nach der Veröffentlichung der Arbeit wurde das Projekt MD5CRK eingestellt.

Diese Attacke wirkt sich nur auf die Kollisionsangriffe aus. Deswegen besteht keine akute Gefahr für Passwörter, die als MD5-Hash gespeichert wurden, diese Kollisionen sind eher eine Gefahr für digitale Signaturen.

6 Pseudocode für den MD5 Algorithmus:

```
1     var int[64] r, k
2     r[ 0..15] := {7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22}
3     r[16..31] := {5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20}
4     r[32..47] := {4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23}
5     r[48..63] := {6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21}

6     k[i] := floor(abs(sin(i + 1)) × 2^32)

7     var int h0 := 0x67452301
8     var int h1 := 0xEFCDAB89
9     var int h2 := 0x98BADCFE
10    var int h3 := 0x10325476

11    var int message_laenge := bit_length(message)
12    erweitere message um bit "1"
13    erweitere message um bits "0" bis Länge von message in bits ≡ 448 (mod 512)
14    erweitere message um message_laenge als 64-Bit little-endian Integer

15    var int a := h0
16    var int b := h1
17    var int c := h2
18    var int d := h3

19    für alle i von 0 bis 63

20    wenn 0 ≤ i ≤ 15 dann
21    f := (b and c) or ((not b) and d)
22          g := i
23    sonst wenn 16 ≤ i ≤ 31 dann
24    f := (d and b) or ((not d) and c)
25          g := (5×i + 1) mod 16
26    sonst wenn 32 ≤ i ≤ 47 dann
27    f := b xor c xor d
28          g := (3×i + 5) mod 16
29    sonst wenn 48 ≤ i ≤ 63 dann
30    f := c xor (b or (not d))
31          g := (7×i) mod 16
32    wenn_ende

33    temp := d
34    d := c
35    c := b
36    b := ((a + f + k(i) + w(g)) leftrotate r(i)) + b
37    a := temp

38    h0 := h0 + a
39    h1 := h1 + b
40    h2 := h2 + c
41    h3 := h3 + d

42    (0  ≤ i ≤ 15): f := d xor (b and (c xor d))
43    (16 ≤ i ≤ 31): f := c xor (d and (b xor c))
```

7 Quellen / Literatur:

- The MD5 Message-Digest Algorithm; R. Rivest, April 1992 / RFC 1321
  - http://www.ietf.org/rfc/rfc1321.txt

- Veröffentlichung des chinesischen Teams über die erste Kollision
  - http://eprint.iacr.org/2004/199.pdf

- Hans Dobbertin, Cryptanalysis of MD5 compress. Announcement on Internet, May 1996

- Hans Dobbertin, The Status of MD5 After a Recent Attack, in CryptoBytes 2(2), 1996

- Wikipedia
  - http://de.wikipedia.org/wiki/Message_Digest_Algorithm_5