Hashfunktionen

Seminar Kryptographische Protokolle
Ismail Demir
Technische Universität München

1 Einführung

1.1 Motivation

Hashfunktionen spielen heute in der Informatik eine wichtige Rolle. Sie finden in mehreren Bereichen Anwendung, die nun im Folgenden kurz erklärt werden, vgl. [1]. In **Datenbanken** werden Hashfunktionen eingesetzt, um schnelle Such- und Zugriffsverfahren für große Datenmengen zu realisieren. Hierfür werden sog. **Hashtabellen** eingesetzt, die eine effiziente Indizierung der einzelnen Einträge ermöglichen. Bei der **Datenübermittlung** und **-sicherung** ist oft die Berechnung von **Prüfsummen** nötig, um fehlerhafte Übertragungen schnell zu erkennen. Um diese Prüfsummen zu berechnen, können Hashfunktionen eingesetzt werden. In der **Kryptologie** werden nicht injektive Hashfunktionen zum **Signieren von Dokumenten** oder zum **Erzeugen von Einweg-Verschlüsselungen** verwendet. Diese auch als **kryptographische Hashfunktionen** bezeichneten Funktionen sind das eigentliche Thema dieser Ausarbeitung.

1.2 Grundsätzliche Begriffe und Definitionen

Bevor wir uns weiter mit den wichtigsten Eigenschaften von Hashfunktionen beschäftigen, werden zunächst einige grundlegende Begriffe definiert.

Definition 1 *Eine Abbildung $h : A \to B$ mit $|A| \geq |B|$ heißt **Hashfunktion**. Der Urbildbereich der Abbildung wird als **Universum** bezeichnet. Der Bildbereich wird als **Adressbereich** bezeichnet. Die Bilder heißen auch **Hashadressen**.*

Als Beispiel betrachten wir eine einfache Hashfunktion, welche die ganzen Zahlen auf Restklassen modulo p abbildet (p sei eine Primzahl). Diese Hashfunktion ist durch die Abbildung

$$\varphi: \begin{cases} \mathbb{Z} & \to p\mathbb{Z} \\ x & \to x + p\mathbb{Z} \end{cases}$$

definiert. Die Menge der ganzen Zahlen ist dabei das **Universum**, die Menge $\varphi(\mathbb{Z}) = \{[0], [1], ..., [p-1]\}$ der **Adressbereich**.

1.3 Kollision und Eindeutigkeit

Eine wichtige Eigenschaft, die für kryptographische Hashfunktionen gefordert wird, ist die **Kollisionsresistenz**. Dafür klären wir zunächst den Begriff der Kollision.

Definition 2 *Sei $h : A \to B$ eine Hashfunktion. Eine **Kollision** tritt auf, falls $a_1, a_2 \in A$ existieren, so dass $a_1 \neq a_2$ und $h(a_1) = h(a_2)$.*

Da Hashadressen in der Kryptologie möglichst eindeutig bzw. kollisionsresistent sein sollen, sind Maßnahmen zur **Kollisionsvermeidung** erforderlich.

2 Sichere Hashfunktionen

2.1 Schwache Hashfunktionen

Die Mindestanforderungen, die eine Hashfunktionen[1] treffen soll, sind durch die folgende Definition gekennzeichnet, vgl. [2, S. 347].

Definition 3 *Es seien A_1 und A_2 endliche Alphabete[2], $k \in \mathbb{N}$, und*

$$H : \begin{cases} A_1^* & \to A_2^k \\ a_1 & \to H(a_1) \end{cases} \text{, wobei } A_1^* := \bigcup_{i \in \mathbb{N}_0} A_1^i$$

*eine nicht injektive Hashfunktion. H heißt **schwache Hashfunktion**, falls die folgenden Bedingungen erfüllt sind.*

1. *Der Hashwert $H(M)$ ist für alle $M \in A_1^*$ leicht zu berechnen (etwa in $O(k^n)$ für $n = 1; 2$), und*
2. *bei gegebenem Hashwert $H(M)$ für ein $M \in A_1^*$ ist es praktisch unmöglich (etwa in $\Omega(e^k)$), ein $M' \in A_1^*$ zu finden, so dass $M \neq M'$ und $H(M) = H(M')$ gilt.*

Es gibt also kein effizientes Verfahren, um zu einer gegebenen Nachricht M eine Nachricht M' zu konstruieren, so dass $H(M) = H(M')$ gilt. Das heißt allerdings nicht notwendig, dass keine Paare (M, M') gefunden werden können, so dass $H(M) = H(M')$ und $M \neq M'$ gilt. Dass sich dadurch ein **Sicherheitsproblem** ergeben kann, zeigt folgendes Beispiel. Betrachten wir ein Szenario, in dem **Signaturen in Kombination mit Hashwerten** eingesetzt werden, um die Urheberschaft von Dokumenten zu beweisen. Der Angreifer konstruiert nun das Nachrichtenpaar (M, M') mit $H(M) = H(M')$ und signiert den Hashwert $H(M)$ für das Dokument M. Wenn er aber später behauptet, seine Unterschrift beziehe sich auf das Dokument M', geht ihre Beweiskraft verloren. Nun stellt sich die Frage, wie schwer es überhaupt ist, solch ein kollidierendes Paar (M, M') zu finden. Das **Geburtstagsparadoxon** kann hierauf eine Antwort geben. Es geht dabei um die Frage, wie viele Personen in einem Raum sein müssen, damit die Chance, dass zwei von ihnen den gleichen Geburstag haben, mindestens 50% ist. Sei K die Anzahl der möglichen Geburstage, d.h. $K = 365$. Dann kann man zeigen, dass näherungsweise $N = \frac{1+\sqrt{1+8K \ln 2}}{2} \approx 23$ Personen für obige Forderung ausreichen. Interpretieren wir K als die Anzahl der möglichen Hashergebnisse, und N als die Anzahl der erforderlichen Tests, dann sind größenordnungsmäßig etwas mehr als $\sqrt{K}$ Tests erforderlich, um ein kollidierendes Nachrichtenpaar zu finden. Da wir stets $A_2 = \{0, 1\}$ und somit $|A_2| = 2$ setzen, gilt $K = 2^k$. Dementsprechend müssen „nur“ noch $2^{k/2}$ Hashwerte berechnet werden, um

[1] Sofern nicht anders angegeben, sind im Folgenden stets kryptographische Hashfunktionen gemeint.

[2] Wir gehen immer von binären Alphabeten aus, d.h. $A_1 = A_2 = \{0, 1\}$.

eine Kollision mit Wahrscheinlichkeit 50% finden zu können. Dieser Prozess wird auch als **Geburtstagsangriff** bezeichnet.

2.2 Starke Hashfunktionen

Es ist nun wünschenswert, dass eine „gute“ Hashfunktion auch dem Geburtstagsangriff standhalten kann. Dieses Ziel wird durch eine Hashfunktion, die der folgenden Definition genügt, erreicht [2, S. 349].

Definition 4 *Sei $H : A_1^* \to A_2^k$ eine schwache Hashfunktion. H heißt **starke Hashfunktion**, falls es praktisch unmöglich ist, ein Paar (M, M') zu finden, so dass $M \neq M'$ und $H(M) = H(M')$ gilt.*

Es muss k also so groß gewäht werden, dass die Berechnung und Speicherung von $2^{k/2}$ Werten praktisch unmöglich ist. Beim heutigen Stand (2009) werden Hashwertlängen von $k \geq 160$ Bits als sicher angesehen. SHA-1 gilt mit einer Hashwertlänge von 160 Bit noch bis Ende 2009, SHA-256 mit 256 Bit noch bis 2012 als sicher [2, S. 349].

2.3 Einsatz sicherer Hashfunktionen

Hashfunktionen werden in der Kryptologie in vielfältiger Weise eingesetzt. Drei der wichtigsten Einsatzgebiete werden wir im Folgenden näher betrachten.

- Bei der **Integritätskontrolle** berechnet der Urheber einer Nachricht M den Hashwert $h := H(M)$ und verschickt diesen Wert zusammen mit M. Der Empfänger des Paares (M', h') berechnet seinerseits den Hashwert $h' := H(M')$, und überprüft h und h' auf Gleichheit. Bei positiver Antwort kann er nun von der Integrität der Nachricht ausgehen, d.h. $M = M'$,vgl. [2, S. 350].

- Beim Verfahren des **signierten Hashwerts**, welches bereits in Kap. 2.1 als Beispiel genannt wurde, wird zu einem Dokument M zunächst der Hashwert $h := H(M)$ und anschließend die Signatur $sig := D(h, S)$ berechnet. Dabei ist S der private Signaturschlüssel und D eine Signaturfunktion. Zuletzt verschickt bzw. hinterlegt der Absender das Paar (M, sig), wodurch der Empfänger die Urheberschaft zweifelsfrei feststellen kann. Der Vorteil gegenüber zum „gewöhnlichen“ Signieren liegt darin, dass nur der relativ kurze Hashwert signiert werden muss - Signieren ist rechenintensiv, vgl. [2, S. 350].

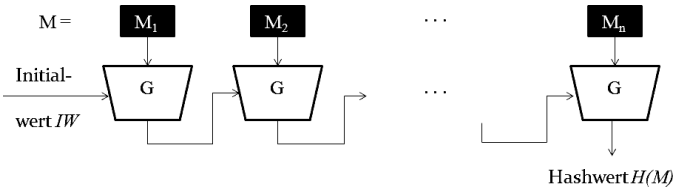
- **Einweg-Verschlüsselungen** kommen in Login-Systemen, bei dem allein der Benutzer sein Passwort kennen soll, zum Einsatz. Nachdem der Benutzer sein Passwort P festgelegt hat, wird lediglich der Hashwert $h := H(P)$ gespeichert. Möchte sich der Benutzer nun mit dem Passwort P' einloggen, wird der Hashwert $h' = H(P')$ berechnet und mit h verglichen. Bei Übereinstimmung kann das System von der Korrektheit des eingegebenen Passworts P' ausgehen, vgl. [3].

2.4 Konstruktion sicherer Hashfunktionen

Hashfunktionen werden gemäß [2, S. 350f.] durch eine Folge (f_i) basierend auf einer Kompressionsfunktion G realisiert. Die Eingabe $M = M_1 M_2 \dots M_n$ wird dabei blockweise iterativ zu einem Hashwert verarbeitet. Alle Blöcke $M_i, i \in \{1, \dots, n\}$ haben die gleiche Größe, wobei der letzte Block ggf. durch Padding aufgefüllt werden muss. Der folgende Algorithmus beschreibt die allgemeine Vorgehensweise.

Algorithmus 5 *Voraussetzung: $M = M_1 M_2 \dots M_n \in A_1^*$ lässt sich in n Blöcke M_i der Länge l zerlegen, und $A := A_1 = A_2$.*

1. *$f_0 := IW$, wobei $IW \in A^k$ ein Initialwert ist.*
2. *$f_i := G(f_{i-1}, M_i)$, wobei $G : A^k \times A^l \to A^k$*
3. *$H(M) := f_n$*



Es lässt sich zeigen, dass eine Hashfunktion gemäß dieser Konstruktion so kollisionsresistent ist, wie die zugrunde liegende Kompressionsfunktion G [4]. Je nach Design von G unterscheidet man zwei Arten von Hashfunktionen, die wir im Folgenden genauer betrachten werden, vgl. [2, S. 351ff.].

2.4.1 Blockchiffren-basierte Hashfunktionen

In diese Kategorie fallen Hashfunktionen, die auf symmetrischen Blockchiffren basieren.

Definition 6 *Eine bijektive Abbildung $E_{KEY} : \{0,1\}^k \to \{0,1\}^k$, $KEY \in \{0,1\}^l$, $k, l \in \mathbb{N}$ heißt **(symmetrische) Blockchiffre** oder **Blockverschlüsselung** mit **Blockgröße** k Bits und **Schlüssellänge** l Bits[3].*

Basierend auf einer Blockchiffre E_{KEY} wird die Kompressionsfunktion G dann wie nachfolgend beschrieben ausgedrückt. Setzen wir beispielsweise die DES-Verschlüsselung für E_{KEY} ein, so ergibt sich die Meyer-Hashfunktion [2, S. 351f.].

$$G : \begin{cases} \{0,1\}^k \times \{0,1\}^l & \to \{0,1\}^k \\ (f_{i-1}, M_i) & \to E_{M_i}(f_{i-1}) \oplus f_{i-1} \end{cases}$$

2.4.2 Dedizierte Hashfunktionen

Hashfunktionen, deren Kompressionsfunktion G speziell für die Erzeugung von Hashwerten konstruiert wurde, heißen dedizierte Hashfunktionen. Sie sind in der Regel deutlich effizienter als Blockchiffren-basierte Hashfunktionen, da kein aufwändiges Verschlüsselungsverfahren durchgeführt werden muss. Die gängigsten Vertreter dieser Kategorie sind die SHA-Familie und das MD5 Verfahren, welche wir uns kurz näher ansehen wollen.

[3] Weitere Informationen über Blockchiffren finden sich in [5].

- Das 1993 vom National Institute of Standards (NIST) veröffentlichte **SHA-1** Verfahren erzeugt 160 Bit Hashwerte und verwendet dabei eine Blockgröße von 512 Bit. Die später entwickelte **SHA-2** Familie beinhaltet die Algorithmen SHA-224, SHA-256, SHA-384, und SHA-512, wobei die angehängte Zahl der jeweiligen Hashwertlänge entspricht. Da 2004 erste Angriffe gegen SHA-1 bekannt wurden, hat das NIST deshalb empfohlen, bis spätestens 2010 auf die SHA-2 Familie umzurüsten [2, S. 352ff.].
- Die MD-Verfahren[4] wurden 1990 von Ronald Rivest eingeführt und 1991 mit **MD5** als einer Weiterentwicklung von MD4 fortgesetzt. Der MD5 liefert einen 128 Bit Hashwert und arbeitet wie SHA-1 auf 512 Bit Blöcken. MD5 gilt heute (Stand 2009) als nicht mehr sicher im Sinne einer starken Hashfunktion. Bereits 2004 konnte von einem chinesischen Wissenschaftlerteam eine erste Kollision gefunden werden [6]. Heute ist MD5 insbesondere nicht mehr geeignet, die Integrität von Daten zu gewährleisten, wie ein beeindruckendes „MD5 Collision Demo“ in [7] zeigt.

3 Message Authentication Codes

Da Hashwerte prinzipiell von jedem berechnet werden können, sind sie nicht geeignet, die Urheberschaft von Dokumenten zu beweisen. Eine Möglichkeit für den Beweis der Urheberschaft besteht in der Benutzung von Message Authentication Codes, vgl. [2, S. 357]. Eine andere Möglichkeit, die wir bereits in Kap. 2.3 kennen gelernt haben, ist die Verwendung von signierten Hashwerten.

Definition 7 *Sei $\{H_K : K \in \kappa\}$ eine Familie, wobei κ ein Schlüsselraum ist, und H_K schwache Hashfunktionen sind. Eine durch diese Familie parametrisierte Hashfunktion heißt* ***Message Authentication Code (MAC)*** *oder* ***Hashfunktion mit Schlüssel****.*

Bemerkung 8 *Statt $H_K(M)$ schreibt man oft auch $H(M, K)$.*

Man beachte, dass der Schlüssel $K \in \kappa$ nur den Kommunikationspartnern bekannt sein darf. Deshalb wird er auch als **MAC-Geheimnis** bezeichnet. Ein wichtiger Punkt ist, dass MACs zwar die Authentizität der Urheberschaft garantieren, nicht jedoch die Authentizität der Daten selbst. Man spricht deshalb von **schwacher Authentizität**. Da stets mehrere Parteien - mindestens zwei - das MAC-Geheimnis kennen, finden MACs keine Verwendung als digitale Unterschrift. Das bedeutet, dass gegenüber einer dritten Partei nicht bewiesen werden kann, welcher der Kommunikationspartner der wahre Absender ist.

3.1 Typische Vorgehensweise

Ein typischer Kommunikationsablauf unter der Benutzung von MACs wird sich im Normalfall wie folgt ereignen.

1. Alice und Bob einigen sich zunächst auf den den Schlüssel K und den MAC H_K.
2. Alice berechnet $m := H_K(M)$ für das Dokument M.
3. Alice verschickt oder hinterlegt das Paar (M, m).
4. Bob erhält das Paar (M', m'), und berechnet $m^* := H_K(M')$.
5. Falls $m' = m^*$ gilt, geht Bob von $M = M'$ und von der Authentizität der Urheberschaft von M' aus.

Das Problem dabei ist die Frage, wie sich Alice und Bob den geheimen Schlüssel K mitteilen. Eine mögliche Lösung besteht in der Verwendung einer digitalen ID einer vertrauenswürdigen Stammzertifizierungsstelle, die beiden Kommunikationspartnern bekannt ist. In einer damit unterzeichneten und verschlüsselten E-Mail könnte Alice Bob das MAC-Geheimnis mitteilen[5].

3.2 MAC-Verfahren

3.2.1 Blockchiffren-basierte MACs

Dieses Verfahren ist dem der blockchiffren-basierten Hashfunktionen sehr ähnlich, jedoch wird das MAC-Geheimnis als Schlüssel verwendet und stattdessen der Rückgabewert der Blockchiffre dann mit dem jeweiligen Block der Nachricht verknüpft. Der entsprechende Algorithmus (ISO 9797) lässt sich wie folgt beschreiben [2, S. 358].

Algorithmus 9 *Voraussetzung: $M = M_1 M_2 \dots M_n \in \{0,1\}^*$ lässt sich in n Blöcke M_i der Länge 64 Bit zerlegen.*

1. $f_1 := M_1$
2. $f_i := E_K(f_{i-1}) \oplus M_i$, *wobei E_K eine Blockchiffre ist.*
3. $MAC(M, K) := E_K(f_n)$

3.2.2 MD5-MAC

Der MD5-MAC liegt einem recht einfachen Verfahren zugrunde, vgl. [2, S. 359]. Zunächst wird die Konkatenation M' aus Nachricht und Schlüssel $M' := M|K$ gebildet, und anschließend die MD5-Hashfunktion darauf angewandt, d.h. $MAC(M, K) = MD5(M')$. Da der Schlüssel selbst jedoch kein integraler Bestandteil des MD5-Verfahrens ist, übertragen sich die bekannten Schwachstellen von MD5 auf den MD5-MAC. Abhilfe verschafft hier das HMAC-MD5 Verfahren, welches zum Schluss noch kurz erläutert wird.

3.2.3 HMAC

Die zentrale Idee des HMAC[6]-Verfahrens besteht darin, den Schlüssel zur Beeinflussung des Initalwerts der Kompressionsfunktion zu verwenden, vgl. [2, S. 359ff.]. Auf diese Weise können die Mängel einer schwachen Hashfunktion beseitigt werden. Der HMAC einer Nachricht M, dem Schlüssel K der Länge r Bytes und der Hashfunktion H, die Blöcke der Länge r Bytes verarbeitet, lässt sich wie folgt berechnen ($r \in \mathbb{N}$).

$$HMAC(M, K) := H(K \oplus opad \mid H(K \oplus ipad \mid M)),$$

[4] Message-Digest Algorithm

[5] Eine solche digitale ID lässt sich z.B. bei [8] erwerben.

[6] keyed-**H**ash **M**essage **A**uthentication **C**ode

wobei *ipad* und *opad* feste Bitfolgen sind[7]. Setzt man für H die MD5-Hashfunktion ein, so spricht man auch vom **HMAC-MD5** Verfahren, welches die Schwachstellen des MD5-MAC beseitigt. Ein Geburstagsangriff auf den HMAC-MD5 würde die Auswertung von 2^{64} Hashwerten erfordern, um die Chance einer Kollision signifikant zu steigern. Weil auch neue Analysen (Stand 2006) keine Angriffsmöglichkeiten auf das HMAC-Verfahren gezeigt haben, gilt dieses Verfahren weiterhin als sicher [2, S. 360].

Literatur

[1] Wikipedia-Autoren, Hashfunktion, Wikipedia, Die freie Enzyklopädie, http://de.wikipedia.org/wiki/Hashfunktion, 24. Mai 2009

[2] Claudia Eckert, IT-Sicherheit: Konzepte, Verfahren, Protokolle, 5. überarbeitete Auflage, R. Oldenbourg-Verlag, 2007

[3] PHP 14.4 Einweg - Verschlüsselung, TEIA AG - Internet Akademie und Lehrbuch Verlag, http://www.teialehrbuch.de/Kostenlose-Kurse/PHP/9438-Einweg-Verschluesselung.html, 24. Mai 2009

[4] Wikipedia contributors, Cryptographic hash function, Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/wiki/Cryptographic_hash_function#Merkle-Damg.C3.A5rd_construction, 24.Mai 2009

[5] Wikipedia-Autoren, Blockverschlüsselung, Wikipedia, Die freie Enzyklopädie, http://de.wikipedia.org/wiki/Blockverschlüsselung, 24. Mai 2009

[6] Wikipedia-Autoren, Message-Digest Algorithm 5, Wikipedia, Die freie Enzyklopädie, http://de.wikipedia.org/wiki/Md5, 24. Mai 2009

[7] Peter Selinger, MD5 Collision Demo, Peter Selinger's Homepage, http://www.mathstat.dal.ca/~selinger/md5collision/, 24. Mai 2009

[8] Digitale IDs für sichere E-Mails, VeriSign Deutschland GmbH, http://www.verisign.de/authentication/individual-authentication/digital-id/index.html, 24. Mai 2009

[7] „ipad“ bzw. „opad“ steht für „inner Padding“ bzw. „outer Padding“