

---

## 5 Digitale Signaturen

Eines der kryptographischen Verfahren, das in der jüngsten Vergangenheit für den meisten politischen Wirbel gesorgt hat, sind digitale Signaturen. Dabei dreht es sich vor allem um die rechtliche Wirkung einer derartigen Signatur.

Mit digitalen Signaturen sollen herkömmliche Unterschriften auf Papierform ersetzt werden. Die eigenhändige Unterschrift wird bereits seit langem als Beweis für die Urheberschaft eines Dokuments oder dem Einverständnis mit seinem Inhalt betrachtet. In diesem Zusammenhang sind folgende Anforderungen von besonderer Bedeutung:

- **Fälschungssicherheit**  
Der Beweis, daß der Unterzeichner und niemand anderes das Dokument unterschrieben hat.
- **Authentizität**  
Für den Empfänger eines Dokuments besteht die Sicherheit, daß der Unterzeichner das Dokument willentlich unterschrieben hat.
- **Integrität**  
Nachdem das Dokument unterzeichnet wurde, kann es nicht mehr unbemerkt verändert werden.
- **Nicht-Wiederverwendbarkeit**  
Die Unterschrift ist Bestandteil des Dokuments und kann nicht auf ein anderes Dokument übertragen werden.
- **Physisches Vorhandensein**  
Sowohl die Unterschrift als auch das Dokument liegen physisch vor, der Unterzeichner kann somit später nicht behaupten, daß er das Dokument nicht unterschrieben hat.

In der Praxis sind diese Anforderungen selten zusammen anzutreffen. So können Unterschriften gefälscht werden, Dokumente wurden unter Zwang unterschrieben, nach der Unterschrift wird dem Dokument noch der ein oder andere Absatz zugefügt und so weiter.

Trotzdem wurde in der Vergangenheit der Ruf nach einem äquivalenten, digitalen Ebenbild immer lauter. Der Einfluß der elektronisch verarbeiteten Dokumente in unserem Alltag wuchs stetig, so daß es nur noch eine Frage der Zeit war, bis auch die Nachfrage nach verbindlichen, durch geeignete kryptographische Verfahren erzeugte digitale Unterschriften größer wurde.

## 5 Digitale Signaturen

Digitale Signaturen basieren in der Regel auf asymmetrischen Verfahren. Dabei wird mit dem privaten Schlüssel unterschrieben und mit dem öffentlichen Schlüssel wird die Signatur auf Echtheit überprüft.

Der sich hierbei ergebende Ablauf kann der folgenden Abbildung entnommen werden.

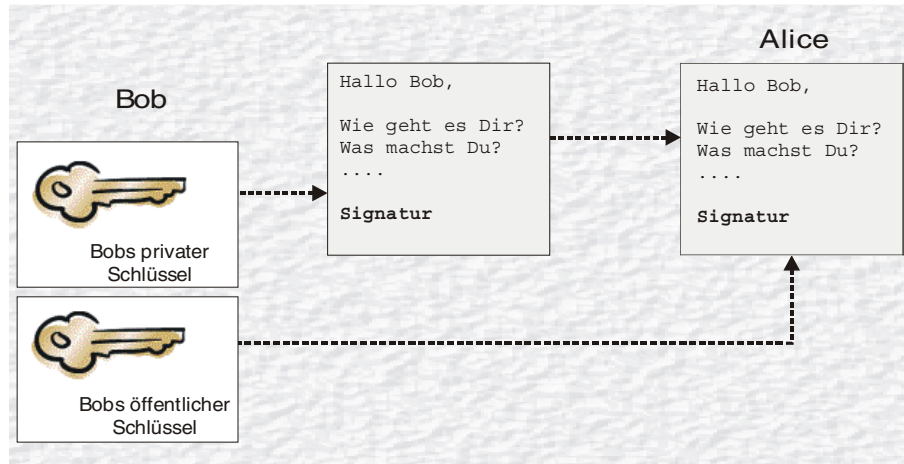


Abb. 5-1 Digitale Signaturen

Bob signiert beziehungsweise verschlüsselt eine Nachricht mit seinem privaten Schlüssel und schickt das Resultat an Alice. Diese überprüft beziehungsweise entschlüsselt sie und kann somit sicher sein, daß die Nachricht von Bob stammt. Bei diesem Ablauf wird die digitale Signatur auf die gesamte Nachricht angewandt. Diese kann somit nicht ohne Kenntnis des privaten Schlüssel von Bob manipuliert werden, so daß die Echtheit der Nachricht sowohl hinsichtlich des Absenders als auch hinsichtlich der Datenintegrität gewährleistet ist.

Soll ein asymmetrisches Verfahren in obigen Sinne genutzt werden, müssen zwei zusätzliche Eigenschaften erfüllt sein: Zum einen müssen Chiffrierung und Dechiffrierung vertauscht werden können und zum anderen muß der Algorithmus für beliebig lange Nachrichten anwendbar sein.

Das asymmetrische Verfahren wird immer auf das gesamte Dokument angewandt, das bedeutet bei größeren Dokumenten, daß die nicht vorhandene Schnelligkeit dieser Verfahren voll zur Geltung kommt. Da eine nur teilweise Anwendung auf das Dokument nicht gewollte rechtliche Konsequenzen nach sich ziehen könnte, erscheint es sinnvoll, die Signatur auf ein Komprimat anzuwenden. Dabei haben sich in der Praxis die sogenannten Einweg-Hashfunktionen durchgesetzt.

## 5.1 Hashfunktionen

Kryptographische Hashfunktionen generieren aus einer beliebig langen Nachricht ein Komprimat im Sinne einer Prüfsumme, die auch als *Hashsumme* oder *Hashwert* bezeichnet wird. Dieser Hashwert ist je nach ausgewähltem Verfahren mit ziemlich großer Sicherheit einzigartig. Selbst zwei Nachrichten, die bis auf ein Zeichen identisch sind, ergeben völlig unterschiedliche Hashsummen.

Hashfunktionen müssen *kollisionsfrei* sein, daß heißt, es soll nicht möglich sein, zwei unterschiedliche Nachrichten mit dem gleichen Hashwert zu erzeugen. Außerdem soll aus einer gebildeten Hashsumme die ursprüngliche Nachricht nicht rekonstruierbar sein. Die Wiederherstellung eines ursprünglichen Textes ist also im Gegensatz zur Chiffrierung nicht mehr möglich, aus diesem Grund werden diese Verfahren auch als »Einweg-Hashfunktionen« bezeichnet.

Im allgemeinen werden Hashfunktionen dazu genutzt, die Manipulationssicherheit von Nachrichten oder sonstigen Daten zu gewährleisten. Dazu wird zuerst ein Hashwert berechnet, der zu einem späteren Zeitpunkt erneut berechnet wird und mit dem ersten Wert verglichen wird. Stimmen beide Werte überein, kann davon ausgegangen werden, daß die Daten nicht manipuliert wurden.

Der Vorteil dieser Verfahrensweise liegt in der Tatsache, daß nicht der gesamte Text, sondern nur ein vergleichsweise geringer Wert je nach Verfahren eine Länge von 128 oder 160 Bit besonders geschützt aufbewahrt oder übermittelt werden muß. Ein Angreifer müßte somit  $2^{64}$  beziehungsweise  $2^{80}$  Dokumente untersuchen, um zwei Nachrichten zu finden, die den gleichen Hashwert haben.

Eine Einweg-Hashfunktion  $H(M)$  generiert aus einer beliebig langen Nachricht einen Hashwert  $h$  fester Länge:

$$h = H(M)$$

Die Nutzung von Hashfunktionen bei digitalen Signaturen läuft vereinfacht wie folgt ab. Alice signiert  $H(M)$  und Bob erzeugt  $H(M')$ . Falls

$$H(M) = H(M')$$

gilt, kann Bob davon ausgehen, daß Alice  $M'$  signiert hat. Den sich hierbei ergebenden Ablauf zeigt die folgende Abbildung.

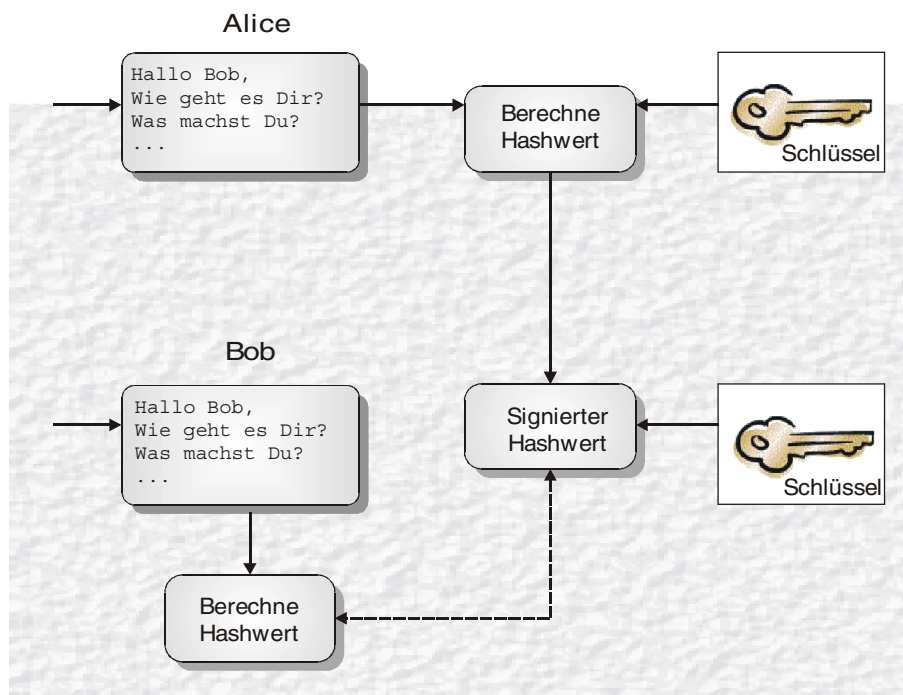


Abb. 5-2 Digital signierter Hashwert

Der generelle Ablauf der Unterzeichnung von Dokumenten mit asymmetrischen Verfahren und Einweg-Hashfunktionen ist wie folgt:

- Zuerst berechnet Alice den Hashwert des Dokuments.
- Anschließend unterzeichnet sie das Dokument mit ihrem privaten Schlüssel.
- Das Dokument übersendet sie zusammen mit dem signierten Hashwert an Bob.
- Bob erstellt mit dem gleichen Verfahren den Hashwert des Dokuments.
- Den signierten Hashwert dechiffriert er mit dem öffentlichen Schlüssel von Alice.
- Stimmen beide Werte überein, ist davon auszugehen, daß die Unterschrift gültig ist.

Diese Verfahrensweise bietet gegenüber der digitalen Signatur ohne Anwendung von Hashfunktionen folgende Vorteile:

- Hashwerte sind kurz, daher kostet die Anwendung von asymmetrischen Verfahren auf diese komprimierten Texte wenig Zeit.

- Dokument und Unterschrift können getrennt gehalten werden. Somit ist eine getrennte, sichere Übermittlung und Verwahrung möglich.
- Hashwerte von Einweg-Hashfunktionen sind nicht voraussagbar, folglich ist kein Angriff mit ausgewähltem Geheimtext möglich.
- Das Dokument ist für jedermann lesbar und bei Kenntnis des öffentlichen Schlüssels jederzeit überprüfbar.

Hashwerte verwenden im allgemeinen keine geheimen Schlüssel, allerdings gibt es auch sogenannte *Message Authentication Codes* (MAC), die die Überprüfung der Hashwerte nur bei Kenntnis des geheimen Schlüssels zulassen. In Bezug auf Hashfunktionen werden diese um einen geheimen Schlüssel  $K$  erweitert, so daß sich

$$h = H(M, K)$$

ergibt. Das bedeutet, daß sich Alice und Bob vorab auf einen gemeinsamen, geheimen Schlüssel  $K$  einigen, diesen in die Berechnung des Hashwertes einbeziehen müssen und der dann, ebenfalls nur unter Einbeziehung des Schlüssels  $K$ , überprüft werden kann. Mallory, der den geheimen Schlüssel nicht kennt, kann somit auch die Korrektheit der Hashsumme nicht verifizieren.

## 5.2 Arbeitsweise einzelner Verfahren

Hashfunktionen haben generell zwei unangenehme Eigenschaften, die direkt miteinander zusammenhängen. Zum einen erweist sich die Entwicklung derartiger Verfahren als relativ komplexe Angelegenheit und daraus resultierend erweist sich mitunter auch die Struktur der Algorithmen als gewöhnungsbedürftig.

Die Komplexität liegt darin begründet, daß Hashfunktionen eine beliebig lange Eingabe akzeptieren müssen und in der Einwegfunktion. Die meisten Verfahren basieren auf einer Kompressionsfunktion, die jeweils die Eingabe eines Textblocks zusammen mit der Ausgabe des vorherigen Textblocks verarbeitet. Der Hashwert lautet dann entsprechend:

$$h_i = f(M, h_{i-1})$$

Der aktuelle Hashwert bildet zusammen mit dem nächsten Textblock die Eingabe für die Kompressionsfunktion. Dieser Vorgang wiederholt sich, bis der letzte Textblock erreicht ist.

Aufgrund der Komplexität von Hashfunktionen werde ich mich im folgenden kurz fassen und lediglich den generellen Ablauf und die wichtigsten Eigenschaften dieser Verfahren aufzeigen.

### 5.2.1 MD5

Das *MD5*-Verfahren (MD steht für »Message Digest«) wurde von Ron Rivest entwickelt und in RFC 1321 spezifiziert.

Das Verfahren verarbeitet Blöcke mit einer Länge von 512 Bit und generiert einen 128-Bit-Hashwert mit folgendem Ablauf:

1. Zuerst wird die Nachricht derart aufgefüllt, daß sie einem Vielfachen von 448 (= 512 – 64) Bit entspricht.
2. Anschließend werden vier Variablen wie folgt initialisiert:  
 $A = 0x01234567$   
 $B = 0x89abcdef$   
 $C = 0xfedcba98$   
 $D = 0x76543210$
3. Beginn der Hauptschleife, die jeweils 512-Bit-Blöcke verarbeitet. Zuerst werden die Variablen aus dem zweiten Schritt nach  $a$ ,  $b$ ,  $c$  und  $d$  kopiert.
4. Der Eingabeblock wird in sechzehn 32-Bit-Teilblöcke unterteilt und es werden vier Runden ausgeführt.
5. Die Variablen  $a$ ,  $b$ ,  $c$  und  $d$  werden jeweils zu  $A$ ,  $B$ ,  $C$  und  $D$  addiert. Das Ergebnis ist der Ausgabeblock, der in der nächsten Runde als Eingabe für  $A$ ,  $B$ ,  $C$  und  $D$  weiter verarbeitet wird.
6. Die Schritte 3, 4 und 5 werden für alle weiteren Eingabeblocke ausgeführt, bis die gesamte Nachricht abgearbeitet ist. Der letzte 128-Bit-Ausgabeblock ist der generierte Hashwert.

Den genauen Ablauf einer einzelnen Runde möchte ich Ihnen ersparen. Nur soviel – In jeder Runde werden mit den binären Funktionen *UND*, *ODER*, *XOR* und *NICHT* Verknüpfungen der Variablen  $a$ ,  $b$ ,  $c$  und  $d$  vorgenommen. Die Operation werden pro Runde sechzehnmal in einer anderen Konstellation genutzt. Außerdem werden verschiedene Additionen, Rotationen, Substitutionen und Multiplikationen mit den einzelnen Bits in verschiedenen nichtlinearen Funktionen ausgeführt.

### 5.2.2 SHA

Der Secure-Hash-Algorithmus (SHA) wurde vom NIST zusammen mit der NSA für den Einsatz mit dem Digital Signature Standard (vergleiche Abschnitt 5.3.2) entwickelt und 1993 veröffentlicht.

Das Verfahren verarbeitet nur Nachrichten mit einer maximalen Länge von  $2^{64}$  Bit. Diese werden ebenfalls in Blöcke mit einer Länge von 512 Bit unterteilt, um einen 160 Bit langen Hashwert zu generieren.

Der Gesamttablauf gestaltet sich folgendermaßen:

1. Die Eingabenachricht wird auf ein Vielfaches von 512 Bit expandiert.
2. Anschließend werden fünf (der Hashwert beträgt im Gegensatz zu MD5 160 Bit) Variablen initialisiert,

$A = 0x67452301$

$B = 0xefcdab89$

$C = 0x98badcfe$

$D = 0x10325476$

$E = 0xc3d2e1f0$

3. Beginn der Hauptschleife, die jeweils 512-Bit-Blöcke verarbeitet. Zuerst werden die Variablen aus Schritt 2 nach  $a$ ,  $b$ ,  $c$ ,  $d$  und  $e$  kopiert.
4. Der 512-Bit-Block wird nach einem bestimmten Verfahren expandiert, in achtzig 32-Bit-Teilblöcke unterteilt und anschließend werden vier Runden ausgeführt.
5. Die Variablen  $a$ ,  $b$ ,  $c$ ,  $d$  und  $e$  werden jeweils zu  $A$ ,  $B$ ,  $C$ ,  $D$  und  $E$  addiert. Das Ergebnis ist der Ausgabeblock, der in der nächsten Runde als Eingabe für  $A$ ,  $B$ ,  $C$  und  $D$  weiter verarbeitet wird.
6. Die Schritte 3, 4 und 5 werden so lange wiederholt, bis alle Eingabeblocke verarbeitet wurden. Der letzte 160-Bit-Ausgabeblock ist der generierte Hashwert.

Wie bei MD5 werden in einer Runde mit den gleichen binären Operationen Verknüpfungen der Eingangsvariablen vorgenommen. Zusätzlich werden bei dem Verfahren noch vier Konstanten  $K_i$  sowie eine Reihe von Operationen und Transformationen verwandt.

### 5.2.3 Sicherheit der Verfahren

Die Ähnlichkeit von MD5 und SHA ist selbst nach diesem Schnellverfahren offensichtlich.

Die folgende Tabelle stellt einige Eigenschaften nochmals zusammen:

	MD5	SHA
Hashwert	128 Bit	160 Bit
Eingabeblock	512 Bit	512 Bit
Anzahl der Schritte	64	80
Maximale Größe einer Nachricht	unendlich	$2^{64}$
Anzahl nichtlinearer Funktionen	4	3
Anzahl Runden/Operationen	4/16	4/20

Aufgrund dieser Angaben ist SHA eine höhere Sicherheit zuzuschreiben, da hier ein höherer Hashwert generiert wird. Zur Erstellung einer Nachricht werden bei einem Brute-Force-Angriff  $2^{160}$  Operationen, bei MD5 dagegen nur  $2^{128}$  Operationen benötigt. Um zwei Nachrichten mit dem gleichen Ergebnis zu erzeugen, sind  $2^{80}$  beziehungsweise  $2^{64}$  Operationen erforderlich.

Somit sollte die Wahl eigentlich zugunsten von SHA ausfallen. Allerdings möchte ich Ihnen noch zwei Hinweise mit auf den Weg geben. Bei der Entwicklung von SHA hatte die NSA ihre Finger im Spiel und dieser Umstand war für einige Hersteller bereits Grund genug, ihre Anwendungen mit MD5 auszustatten.

Aber auch bei diesem Verfahren wurden mittlerweile Schwachstellen gefunden. Hans Dobbertin zeigte 1996 eine Schwachstelle in der Kompressionsfunktion auf und wie Kollisionen berechnet werden können [Dob96]. MD5 gilt seitdem nur noch als bedingt sicher, was in diesem Zusammenhang bedeutet, daß auf der einen Seite von der Nutzung abgeraten wird und auf der anderen Seite steht die Aussage einiger Experten, daß die gefundene Schwachstelle keinerlei Auswirkungen auf die Nutzung des Verfahrens in der Praxis habe.

### 5.2.4 Weitere Verfahren

Obwohl sich im Bereich der Hashfunktionen fast alles um MD5 oder SHA-1 dreht, kann es passieren, daß Ihnen der eine oder andere Algorithmus in der Praxis begegnet. Dabei kann es sich unter anderem um die folgenden Verfahren handeln:

- MD2

Eine weitere von Ron Rivest entwickelte Einweg-Hashfunktion. Das Verfahren ist sehr langsam, aber im Gegensatz zu den anderen Hashfunktionen relativ einfach aufgebaut. Falls Sie sich näher mit Design und Funktionsweise dieser Verfahren beschäftigen wollen, finden Sie unter RFC 1319 eventuell den passenden Einstieg.

- MD4

Das Verfahren wurde von Ron Rivest entworfen und in RFC 1320 näher beschrieben. Der Algorithmus konnte bereits kurz nach der Veröffentlichung erfolgreich kryptanalytisch analysiert werden. Obwohl sich die Angriffe nur auf einzelne Runden bezogen, verbesserte Rivest das Verfahren und nannte es »MD5«.

- RIPE-MD160

Das Verfahren wurde im Rahmen des EU-Projekts RIPE (RACE Integrity Primitives Evaluation, 1988-1992) entwickelt und erzeugt einen 160-Bit-Hashwert. Das Verfahren gilt als sehr sicher, es wurden bislang keinerlei Angriffe bekannt. RIPE-MD160 wartet dennoch noch auf den internationalen Durchbruch (beziehungsweise in den USA). Das Verfahren ist mindestens als sinnvolle Alternative zu MD5 und SHA zu sehen.

## 5.3 Digitale Signaturen

Digitale Signaturen können sowohl mit symmetrischen als auch mit asymmetrischen Verfahren erzeugt werden. Obwohl sich die erste Variante als die einfachere herausstellt, werden in der Praxis dennoch – fast – nur Public-Key-Algorithmen zum Signieren genutzt. Das folgende Beispiel soll diesen Sachverhalt verdeutlichen.

Zuerst müssen Alice und Bob einen geheimen Schlüssel vereinbaren, mit dem sie Dokumente signieren wollen. Nachdem Alice ein Dokument signiert hat, kann Bob nach Erhalt der Nachricht die Unterschrift mittels des gemeinsamen, geheimen Schlüssels überprüfen. Allerdings ist dazu nur Bob in der Lage, es sei denn, daß der Schlüssel in einem Rundschreiben noch weiteren Kommunikationspartnern bekannt gegeben wurde. Eine Veröffentlichung kommt aus naheliegenden Gründen

ebenfalls nicht in Betracht. Außerdem kann bei dieser Verfahrensweise Bob jederzeit die Unterschrift von Alice fälschen und umgekehrt.

Symmetrische Verfahren eignen sich, wenn überhaupt, nur bei der wiederholten Anwendung bei einem kleineren Kreis von Teilnehmern. Deshalb wird in der Regel auf asymmetrische Verfahren zugegriffen.

Da sich in Deutschland selten etwas unbürokratisch regeln läßt, was weitreichende Auswirkungen haben könnte, wurde auch im Bereich der digitalen Signaturen eine gesetzliche Grundlage geschaffen.

### 5.3.1 Das deutsche Signaturgesetz

Das deutsche Signaturgesetz (SigG) gibt es seit dem ersten Juli 1997. Es regelt die Genehmigung von Zertifizierungsstellen, die die Zuordnung von öffentlichen Schlüsseln zu natürlichen Personen bescheinigen.

Das SigG bezieht sich ausschließlich auf Public-Key Verfahren und spricht in diesem Zusammenhang von einer elektronischen statt einer digitalen Signatur. Im Grunde genommen soll dies als offener Ansatz des Gesetzgebers verstanden werden und weitere Verfahren mit einschließen. Da es derzeit aber noch keine weiteren Verfahren gibt, deckt die digitale Signatur die elektronische ab.

Derzeit werden nur Zertifikate von natürlichen Personen abgedeckt, das heißt, die Zertifizierung von juristischen Personen (Kapital- und Personengesellschaften) oder Gruppen von natürlichen Personen sind nach dem SigG ebenso ausgeschlossen wie die direkte Zertifizierung von maschinell erzeugten Zertifikaten, also beispielsweise SSL oder SET. Darüber hinaus ist auch die Zertifizierung von PGP-Schlüsseln nicht SigG-konform.

Sinn und Zweck des SigG ist es, Rahmenbedingungen für digitale Signaturen zu schaffen, unter denen diese als sicher gelten und Fälschungen digitaler Signaturen oder Manipulationen von signierten Daten zuverlässig festgestellt werden können.

Der deutsche Gesetzgeber würde seinem Ruf nicht gerecht werden, wenn er es bei einer Richtlinie belassen würde. Da sich das SigG auch auf das Internet auswirken würde und dieses in der jüngsten Vergangenheit noch als rechtsfreier Raum anzusehen war, wurde noch das sogenannte »Multimediasgesetz« verabschiedet, das einheitliche wirtschaftliche Rahmenbedingungen für die verschiedenen Möglichkeiten der Informations- und Kommunikationsdienste schaffen soll. Dieses Gesetz war das weltweit erste, das sich speziell mit Online-Diensten befaßte. Die offizielle Bezeichnung hierfür lautet »Informations- und Kommunikationsdienste-Gesetz« (IuKDG).

Im IuKDG werden Fragen des Urheberrechts, des Jugend- und Datenschutzes sowie der Nutzung von Telediensten geregelt. Auf diese Weise soll ein Schutz für persönliche und geschäftliche Daten aufgebaut werden.

Das Signaturgesetz wird noch durch zwei weitere Regelungen ergänzt: Zum einen durch die Signaturverordnung (SigV), in der konkrete Vorschriften für Zertifizierungsstellen zu finden sind, und zum anderen die Richtlinie des Europäischen Parlaments und des Rates über gemeinschaftliche Rahmenbedingungen für elektronische Signaturen, die im Januar 2001 in Kraft gesetzt wurde. Diese Richtlinie hat aufgrund des geltenden EU-Rechts keine Direktwirkung. Anders ausgedrückt bedeutet dies, daß es keinen neuen EU-Standard zusätzlich zu den jeweils geltenden nationalen Signatur-Standards geben wird. Die Richtlinie wird demnach jeweils auf nationaler Ebene von den einzelnen Mitgliedsstaaten umgesetzt.

Insgesamt läßt sich der deutsche Gesetz- und Richtlinien-Dschungel wie folgt zusammenfassen:

- Das Signaturgesetz regelt die Form der Zertifikate und legt die Voraussetzungen fest, die die Zertifizierungsstellen erfüllen müssen.
- Als staatliche Behörde für die Lizenzierung und die Überprüfung der Zertifizierungsstellen ist die Regulierungsbehörde im Fachbereich des Bundeswirtschaftsministeriums vorgesehen. Dies ist die Regulierungsbehörde für Telekommunikation und Post (RegTP).
- Der Zertifizierungsstelle ist ausdrücklich verboten, private Schlüssel zu speichern. Damit ist ein Mitlesen staatlicher Stellen zur Strafverfolgung verschlüsselter Informationen nicht möglich.
- Ob die Überwachung verschlüsselter Texte dem Staat in anderer Form gewährleistet wird, ist Gegenstand einer Verschlüsselungsdebatte, die gegenwärtig in einigen Industriestaaten geführt wird.

Seitens des deutschen Gesetzgebers wurden somit die Voraussetzungen zur Nutzung der digitalen Signatur neben der handschriftlichen Unterschrift geschaffen. Dennoch werden Sie diese in der Praxis lange suchen, bevor Sie fündig werden. Zertifizierungsstellen sind ebenso wie Anwendungen, die das Verfahren umsetzen, relativ dünn gesät. Die Gründe sind ebenso trivial wie schwierig, die gesetzlichen Grundlagen lassen nur einen geringen Freiraum übrig. Was auf den ersten Blick vernünftig erscheint – schließlich wollen Sie sich ja abgesichert fühlen – ist auf den zweiten Blick nur schwer praktisch umzusetzen.

Die im Signaturgesetz geforderten Voraussetzungen für Zertifizierungsbehörden sind nicht so ohne weiteres mal eben zu schaffen. Der Rahmen ist sehr eng gestrickt und viele Unternehmen scheuen den damit verbundenen Aufwand.

Dieser Zustand sollte sich mit der besagten EU-Richtlinie und der daraus resultierenden Novellierung der nationalen Vorschriften ändern. Der deutsche Bundesrat verabschiedete im Februar 2001 eine Neufassung des Signaturgesetzes, die so gut wie keine technische Vorschriften enthält und Zertifizierungsstellen selbst für entstandene Schäden haftbar macht. Allerdings sind nach wie vor strenge Auflagen an die einzelnen Zertifizierungsstellen gestellt.

Die von der Bundesregierung als »Durchbruch« gefeierte Novellierung stößt in der Wirtschaft weiterhin auf eine zögernde Zurückhaltung. Eine Beurteilung, inwieweit sich diese Haltung in der Zukunft ändern wird, erweist sich als schwierig. Auf der einen Seite hat der Gesetzgeber die rechtlichen Voraussetzungen zum Schutz der Bürger zu veranlassen, auf der anderen Seite stehen einzelne Unternehmen, die der Meinung sind, daß die Regierung ihre Maßnahmen in diesem Bereich etwas übertrieben habe.

### 5.3.2 Digital Signature Standard

Bereits 1991 wurde vom NIST ein Vorschlag für einen Digital Signature Algorithm (DSA) für die Nutzung im Digital Signature Standard (DSS) veröffentlicht. Mit diesem Standard wurde ein asymmetrisches Verfahren, besagter DSA, für die Überprüfung digitaler Signaturen, der Integrität von Daten und der Authentifizierung des Absenders spezifiziert.

Nach der Veröffentlichung wurde insbesondere der zugrunde liegende Algorithmus eifrig kritisiert. Da zu dieser Zeit RSA noch unter Patentschutz stand und das Verfahren in zahlreichen Anwendungen implementiert war, ging es bei der Argumentation vorrangig um finanzielle Gründe. RSADSI fürchtete um den Verlust von Einnahmen; Unternehmen, die den RSA-Algorithmus in ihren Produkten nutzten, sahen die Gefahr, diese auf ein neues Verfahren umstellen zu müssen. Vereinzelt Anwender dagegen lehnten den Standard ab, da der Algorithmus von der NSA entwickelt wurde. Obwohl das Verfahren öffentlich bekannt gemacht wurde und somit für eine Analyse zur Verfügung stand, ließ sich dieser Punkt nicht aus der Welt schaffen. Wir werden darauf noch zu sprechen kommen (vergleiche 5.6.2).

Allen Kritiken zum Trotz wurde der Standard letztlich drei Jahre später festgelegt. Die Funktionsweise entspricht im wesentlichen einer Variante des ElGamal-Verfahrens und wird an dieser Stelle nicht weiter besprochen.

## 5.4 Anwendung digitaler Signaturen

Im folgenden Abschnitt wird der prinzipielle Ablauf bei der Erzeugung einer digitalen Signatur und der sich anschließenden Verifizierung dargestellt. Dazu wird zuerst der Hashwert für das zu signierende Dokument berechnet.

### 5.4.1 Erzeugen eines Hashwerts

Wie bereits erwähnt, empfiehlt es sich bei größeren Nachrichten zuerst einen Hashwert zu berechnen und diesen anschließend zu signieren. Als Beispiel wird das folgende Dokument genutzt, das unter »moonraker.txt« abgespeichert ist.

Bond im All: Eine Boeing 747, die mit einem US-Space-Shuttle beladen ist, stuerzt in den Atlantik. Als die Englaender das Wrack bergen wollen, finden sie keine Spur des Weltraum-Pendlers. Der Geheimdienst beauftragt James Bond, sich den Hersteller, Drax-Industries, einmal genauer anzusehen. Er stoest auf eine moerderische 'außerirdische' Bedrohung ...

Bei der Erzeugung der Hashsumme wird einmal mehr auf Perl-Module zugegriffen, die Berechnung eines Hashwertes einer Datei kann beispielsweise wie folgt erfolgen:

```
#!/usr/bin/perl -w

use Crypt::RIPEMD160;

$md = new Crypt::RIPEMD160;

$file = "moonraker.txt";
open(FILE, $file) or die "Fehler beim Oeffnen '$file': $!";
binmode(FILE);

$md->addfile(FILE);
$hexd = $md->hexdigest;

print "Hashwert: $hex\n";
```

*Listing 5-1 – ripemd160.pl*

An und für sich nichts weiter neues – zuerst wird das Verfahren eingebunden (hier *RIPEMD160*), die Datei aufgerufen und mit der Methode *addfile()* werden die Daten übergeben. Anschließend wird der Hashwert generiert und ins Hex-Format umgewandelt, bevor es letztlich zur Ausgabe kommt:

```
$ ./ripemd160.pl  
Hashwert: fdabc8fa 1c4dfdda 629ab0c1 1186e13d 15e8890f  
$
```

Wie bereits gesagt, führt die kleinste Änderung einer Datei zu einem völlig anderen Hashwert. Da diese Aussage derzeit noch im Raum steht, wird an das Ende der Datei eine weiterer Punkt eingefügt. Ein erneuter Aufruf führt zu folgender Ausgabe:

```
$ ./ripemd160.pl  
Hashwert: 6b61bd52 11d05a3f 821ccb1b 0827e2ea 9abccc45  
$
```

Dieser Aufruf zeigt deutlich, daß die Manipulation eines einzelnen Zeichens bereits dazu führt, daß ein komplett anderer Hashwert generiert wird. Somit ist also sichergestellt, daß jegliche Änderung bei der erneuten Erzeugung einer Hashsumme sofort auffällt.

Nachdem der Hashwert erzeugt wurde, gilt es, diesen digital zu signieren, was beispielhaft am ElGamal-Verfahren gezeigt wird.

### 5.4.2 Signieren mit ElGamal

Wie bei asymmetrischen Verfahren üblich, ist zuerst ein Schlüsselpaar zu erzeugen. Dazu sind zuerst eine Primzahl  $p$  und zwei weitere Zahlen  $g$  und  $x$ , die beide kleiner als  $p$  sein müssen, zu wählen. Mit diesen Werten ist

$$y = g^x \text{ mod } p$$

zu berechnen. Der öffentliche Schlüssel besteht aus den Werten  $y$ ,  $g$  und  $p$ , der private Schlüssel aus  $x$ .

Um eine digitale Signatur der Nachricht  $M$  zu erzeugen, wird eine weitere Zufallszahl  $k$  gewählt, die teilerfremd zu  $(p - 1)$  ist. Anschließend wird

$$a = g^k \text{ mod } p$$

berechnet und die folgende Gleichung nach  $b$  aufgelöst:

$$M = (xa + kb) \text{ mod } (p - 1)$$

Die Signatur besteht aus den Werten von  $a$  und  $b$ ,  $k$  ist geheim zu halten.

Die Überprüfung der Signatur erfolgt durch

$$y^a a^b \text{ mod } p = g^M \text{ mod } p$$

Hierbei ist ebenfalls darauf zu achten, daß für jede digitale Signatur eine neue zufällige Zahl  $k$  verwendet wird, damit es einem Angreifer nicht möglich ist, aus

zwei Nachrichten die ein gleiches  $k$  verwenden, den privaten Schlüssel  $x$  herauszufinden.

Das Verfahren scheint etwas unübersichtlich, ist aber nicht allzu schwierig, wie das folgende Beispiel zeigt. Mit  $p = 17$ ,  $g = 13$  und  $x = 9$  wird zuerst  $y$  berechnet.

$$y = g^x \text{ mod } p$$

$$\begin{aligned} y &= 13^9 \text{ mod } 17 \\ &= 13 \end{aligned}$$

Um die Nachricht  $M = 8$  (der Hashwert aus dem vorigen Beispiel erschien mir an dieser Stelle etwas zu unhandlich) zu signieren, wird eine weitere Zufallszahl  $k = 7$  gewählt und  $a$  berechnet.

$$a = g^k \text{ mod } p$$

$$\begin{aligned} a &= 13^7 \text{ mod } 17 \\ &= 4 \end{aligned}$$

Letztlich ist noch die folgende Gleichung nach  $b$  aufzulösen:

$$M = (xa + kb) \text{ mod } (p - 1)$$

$$8 = (9 \cdot 4 + 7 \cdot b) \text{ mod } (17 - 1)$$

$$12 = b$$

Was letztlich zu  $b = 12$  führt, dieser Wert und  $a$  bilden die Signatur der Nachricht  $M$ .

Bob kann nach Erhalt der Nachricht die Signatur durch Übereinstimmung der linken und der rechten Seite gemäß

$$y^a a^b \text{ mod } p = g^M \text{ mod } p$$

$$13^4 4^{12} \text{ mod } 17 = 13^8 \text{ mod } 17$$

$$1 = 1$$

überprüfen. Scheint also zu stimmen.

### 5.5 Sicherheit digitaler Signaturen

Nachdem bereits auf die Sicherheit der Verfahren zur Erzeugung von Hashfunktionen eingegangen wurde, soll an dieser Stelle aufgezeigt werden, welche Möglichkeiten ein Angreifer im Zusammenhang mit digitalen Signaturen hat.

Das Hauptziel von Mallory oder jedes anderen Angreifers besteht darin, eine gültige Unterschrift unter ein weiteres Dokument zu setzen. Dazu kann er entweder ein zweites Dokument erstellen, das den gleichen Hashwert liefert, oder er findet den privaten Schlüssel von Alice heraus, der ihn in die Lage versetzt, jedes beliebige Dokument zu signieren.

Außer den üblichen Möglichkeiten wie dem Knacken des privaten Schlüssels oder Berechnung der Umkehrfunktion bieten digitale Signaturen noch einige weitere Varianten. Nur so am Rande sei hier noch erwähnt, daß die Kompromittierung der privaten Schlüssel dadurch ausgeschlossen werden kann, daß der Schlüssel nach jeder Signatur vernichtet wird. Zur Überprüfung wird ja nur der öffentliche Schlüssel benötigt. Der Nachteil liegt natürlich darin, daß für jede Signatur ein neues Schlüsselpaar erzeugt werden muß. Abgesehen von dem damit verbundenen Aufwand, kann naturgemäß die Anzahl der Schlüssel im Laufe der Zeit zu einer gewissen Unübersichtlichkeit führen.

#### 5.5.1 Der Geburtstagsangriff

Der *Geburtstagsangriff* zielt auf Einweg-Hashfunktionen ab, die einen geringen Hashwert produzieren. Angenommen, Mallory will vortäuschen, daß Alice anstelle des ursprünglichen Vertrags einen anderen unterzeichnet, der sich für ihn »günstiger« auswirkt. Also beispielsweise wenn einer Summe, die Alice an ihn zu leisten hat, von ursprünglich einhundert auf tausend EUR geändert wird.

Nehmen wir weiter an, der Hashwert hat eine Länge von 20 Bit, dann könnte Mallory wie folgt vorgehen: Zuerst ändert er die Summe in seinem Sinne, anschließend sucht er sich eine unverfängliche Stelle im Text und fügt beispielsweise ein Leerzeichen ein. Von diesem Dokument berechnet er erneut den Hashwert. Stimmen beide überein, ist er am Ziel, wenn nicht, ändert er das Dokument erneut, bis beide Hashsummen übereinstimmen. Berechnet Mallory ausreichend viele Hashwerte, liegt die Wahrscheinlichkeit, den »richtigen« Wert zu ermitteln, gar nicht so schlecht.

Zum Schutz vor derartigen Angriffen berechnet Alice den Hashwert neu und verdoppelt dessen Länge auf 40 Bit. Alice fühlt sich jetzt besser, da Mallory weitaus mehr Dokumente erzeugen und deren Hashwerte ermitteln muß. Aber weit gefehlt, tatsächlich kann Mallory mit dem gleichen Aufwand zum Ziel kommen. Dazu erstellt er die jeweils gleiche Anzahl von »korrekten« und ihn begünstigenden

Dokumenten und berechnet anschließend die Hashsummen. Die Wahrscheinlichkeit, ein Dokumentenpaar mit dem gleichen Hashwert erzeugt zu haben, ist hierbei weitaus größer als zu einem gegebenen Dokument die betreffende Anzahl getürkter Dokumente zu erstellen. Eine relativ hohe Trefferquote erzielt Mallory, wenn er jeweils  $2^{20}$  Dokumente erstellt, dann muß er nur noch Alice das Dokument signieren lassen und die Unterschriften austauschen.

Warum heißt diese Angriffsart aber Geburtstagsangriff? Die Bezeichnung beruht auf einem Standardproblem der Statistik. Wie groß muß eine Gruppe von Personen sein, damit mit einer Wahrscheinlichkeit von mehr als fünfzig Prozent eine der Personen am gleichen Tag Geburtstag hat wie der Gastgeber? Die Antwort hierzu lautet 253. Allerdings müssen nur 23 Personen zusammen sein, um mit einer Wahrscheinlichkeit von mehr als fünfzig Prozent zwei Personen zu finden, die am gleichen (aber einem beliebigen) Tag Geburtstag haben.

Welche Möglichkeiten hat Alice, diese Angriffe abzuwehren? Zum einen, indem ausreichend große Hashwerte erzeugt werden, und zum anderen sollte ein Algorithmus genutzt werden, der keinerlei Kollisionen erzeugt.

### 5.5.2 Verdeckte Kanäle

Gustavus Simmons entwickelte eine Methode, mit der ein *verdeckter Kanal* in einem Verfahren für digitale Signaturen untergebracht werden kann. Dabei handelt es sich um die Möglichkeit, in einer Signatur weitere verborgene Mitteilungen zu verstecken, ohne daß einem Außenstehenden irgend etwas an der Signatur auffällt.

Das Verfahren läuft im allgemeinen wie folgt ab:

- Alice erzeugt eine beliebige Nachricht, die sie mit einem mit Bob vereinbarten geheimen Schlüssel unterzeichnet.
- Diese unterschriebene Nachricht, die eine verdeckte Mitteilung in der Signatur enthält, übersendet sie Bob.
- Bob überprüft die Signatur, um sicherzugehen, daß die Nachricht auch wirklich von Alice stammt.
- Anschließend extrahiert Bob die verdeckte Nachricht mit dem gemeinsamen geheimen Schlüssel, der eigentliche Inhalt der Nachricht ist bei diesem Verfahren von geringerem Interesse.

Bei dem geheimen Schlüssel handelt es sich je nach genutzten Verfahren um den privaten Schlüssel von Alice oder einen weiteren Schlüssel.

Mallory hat bei diesem Verfahren gleich mehrere Probleme: Er kann keine versteckten Mitteilungen hinzufügen, er kann die verdeckte Nachricht nicht lesen und – was das wichtigste hierbei ist – er kann der Nachricht nicht ansehen, daß in

– was das wichtigste hierbei ist – er kann der Nachricht nicht ansehen, daß in der Signatur eine weitere, verdeckte Mitteilung enthalten ist. Etwas problematischer sieht die ganze Sache zwischen Alice und Bob aus. Je nach Verfahrensart müssen die Kommunikationspartner darauf vertrauen, daß der private Schlüssel nicht zum Nachteil des jeweils anderen Teilnehmers genutzt wird.

Was hat das ganze aber mit einem Angriff zu tun? Verdeckte Kanäle lassen sich bei mehreren Signaturverfahren erzeugen. Besondere Beachtung verdient in diesem Zusammenhang der von der NSA entwickelte Algorithmus DSA, der verdeckte Kanäle bietet, die auch ohne Kenntnis des privaten Schlüssels gelesen werden können. Weder das NIST noch die NSA wußten angeblich etwas von dieser Möglichkeit.

Angenommen, Sie erwerben eine Anwendung für digitale Signaturen und der Algorithmus verfügt über derartige Kanäle. Dann hat der Hersteller der Anwendung die Möglichkeit, jedesmal wenn Sie eine Signatur erzeugen, ein paar Bits in den versteckten Kanal einzulesen. Verfäht das Unternehmen oder ein Verbündeter auf diese Weise, haben diese über kurz oder lang Ihren privaten Schlüssel in ihrem Besitz und sind somit in der Lage, Ihre Signatur zu fälschen, Sitzungsschlüssel zu berechnen, Nachrichten mitzulesen und ähnliches. Der erzeugten Signatur selbst ist nicht anzumerken, daß sie noch eine derartige Zusatzfunktion ausführt.

Als Anwender haben Sie generell zwei Möglichkeiten, sich gegen das Ausspionieren Ihres privaten Schlüssels zu schützen. Die erste Variante ist, daß Sie beispielsweise auf PGP zurückgreifen, das von der Internetgemeinde auf Herz und Nieren überprüft wurde. Bislang wurden darin keinerlei verdeckten Kanäle oder sonstige Varianten entdeckt.

Die zweite Möglichkeit ist, den Kanal einfach zuzumachen. Dazu muß der Kanal logischerweise bekannt sein, womit wir wieder bei dem allgemeinen »Gut-und-Böse-Syndrom« sind. Sind bei Verfahren XY auch wirklich alle Kanäle bekannt? Ist die ergriffene Maßnahme auch wirklich sicher? Und so weiter. Prinzipiell erinnert das ein wenig an die generelle Frage nach der Sicherheit eines Verfahrens oder einer Anwendung. Entweder es wird dem Verfahren vertraut oder eben nicht. Anders ausgedrückt: Sie müssen bei der Auswahl eines Signaturverfahrens noch einen weiteren Punkt bezüglich der Sicherheit des Verfahrens beachten. Das Zumachen eines Kanals erweist sich in der Regel als schwieriger, als es auf den ersten Blick vielleicht scheint. Derartige Maßnahmen sind vom jeweiligen Verfahren abhängig und eine ausführliche Darstellung würde den Rahmen hier bei weitem übersteigen.

## 5.6 Zusammenfassung

Die digitale Signatur hat im Gegensatz zur handschriftlichen Unterschrift eine Reihe von Problemen zu bewältigen. Dabei steht unangefochten an erster Stelle die Authentizität. Ein Fälscher mag noch so geschickt die Schwingungen Ihrer Unterschrift nachahmen können, einem versierten Graphologen werden dennoch irgendwelche Unterschiede auffallen, so daß der Schwindel letztlich entdeckt werden kann.

Bei dem digitalen Gegenstück sieht es dabei schon anders aus, denn individuelle Eigenheiten lassen sich hierbei nicht erkennen. Sobald der private Schlüssel in fremden Besitz ist, haben Sie verloren, eine Fälschung kann nicht mehr vom Original unterschieden werden und letztlich steht Aussage gegen Aussage.

Aber selbst ein hundertprozentig sicheres Verfahren schafft nicht alle Probleme aus der Welt. Auch die Identifizierung des Signierenden kann sich mitunter, wenn auch in geringerem Maße, als Problem erweisen. Hierbei gibt es je nach Anwendung unterschiedliche Lösungsansätze. Sie sollten diesen Punkt jedoch nie völlig aus den Augen verlieren.

Zu guter Letzt noch ein weiteres Problem. Wenn Sie einen Kaufvertrag über Ihr Wochenendgrundstück auf den Bahamas unterschreiben, haben Sie den Vertrag vor sich. Bei einer digitalen Signatur kann das anders sein. Besonders beim Signieren eines Hashwertes ist besondere Vorsicht geboten, aber auch sonst kann Ihnen ein anderes Dokument untergeschoben werden.

Aus diesen Gründen ist auch ersichtlich warum sich die digitale Signatur bislang nicht hat durchsetzen können. Die Abwehrhaltung potentieller Anwender begründet sich einfach in der Tatsache, daß es noch zuviele Wenn und Aber gibt. Der Gesetzgeber hat zwar, zumindest aus seiner Sicht, die notwendigen Voraussetzungen geschaffen, aber mit einer Akzeptanz von heute auf morgen war in diesem Zusammenhang auch nicht zu rechnen. Bleibt nur abzuwarten was die Zukunft für digitale Signaturen bereit hält.