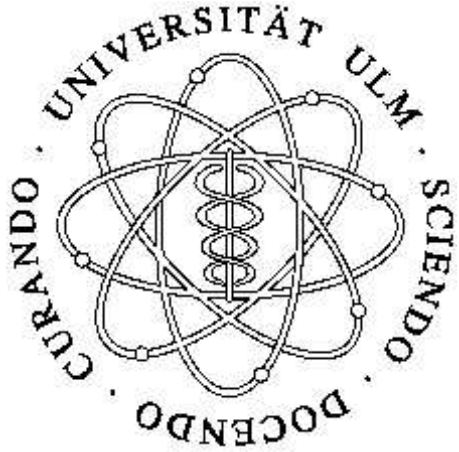
Digitale Signaturen
- Theorie und Praxis -

Diplomarbeit an der Universität Ulm
Fakultät für Informatik



vorgelegt von:

Silke Bergmaier

1. Gutachter: *PD Dr. Johannes Köbler*
2. Gutachter: *Prof. Dr. Franz Schweiggert*

1999

VORWORT

Mit der Verabschiedung des Digitalen Signaturgesetzes im August 1997 wurde zum ersten Mal der rechtliche Rahmen für den Einsatz Digitaler Signaturen geschaffen. Anfang des Jahres 1998 startete die Regulierungsbehörde für Telekommunikation und Post ihre technische Betriebsaufnahme mit 3000 Mitarbeitern mit dem Ziel, alles zu „überwachen", was mit der Einhaltung des Signaturgesetzes im Zusammenhang steht. Seit dem 24. September 1998 gilt die Regulierungsbehörde zudem als das oberste deutsche „Trust Center", welches berechtigt ist, andere „Trust Center" für Digitale Signaturen zu akkreditieren.

Nun, zwei Jahre nach Verabschiedung des Gesetzes, trat am 1. Januar 1999 das erste von der Regulierungsbehörde genehmigte Trust Center, die Telekom ihren Dienst an. Ihre Aufgabe besteht u.a. darin, weitere Unternehmen als Zertifizierungsstellen für Digitale Signaturen zu akkreditieren, um damit die Verbreitung Digitaler Signaturen im alltäglichen Leben zu fördern.

Diese Arbeit soll im folgenden zunächst darüber Aufschluß liefern, wie Digitale Signaturen erstellt werden, welche Sicherheitsvorkehrungen dabei getroffen werden müssen und wie diese umgesetzt werden können. Daraufhin werden praktische Einsatzbereiche digitaler Signaturen geschildert und deren rechtliche Grundlage erörtert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 0

Einleitung

Das Internet-Zeitalter hat begonnen: Mittlerweile stellen die Begriffe „Homebanking", „Elektronische Mails" oder „Einkaufen im Internet" selbst für die breiten Massen der Gesellschaft keine Fremdwörter mehr dar. Der digitale[1] Datenaustausch ist auf wirtschaftlicher Ebene mittlerweile unverzichtbar geworden: Sei es nun innerbetrieblich zum Austausch wichtiger Produkt- oder Personen-Daten oder auch interkommunikativ zwischen Firmenfilialen oder Behörden. Nirgendwo sonst können Daten so schnell und kostengünstig übermittelt werden, wie über das Internet. Aber auch im privaten Bereich gewinnt der digitale Datenaustausch mehr und mehr an Bedeutung: Sei es zur Abwicklung von Banktransaktionen wie z.B. Überweisungen mittels PC, sei es zur Beschaffung von Informationen über das WWW [2] oder auch „nur" zum persönlichen Nachrichtenaustausch via E-Mail. Der digitale Datenaustausch spart vor allem durch schnellere Übermittlungszeiten, als dies z.B. per Post möglich wäre, viel Zeit und damit auch Geld.
Dabei stellt sich jedoch die Frage, inwieweit man dieser Datenübertragung Vertrauen schenken kann. Wer kann dafür garantieren, daß nicht Daten während der Übertragung abgehört, verändert bzw., daß nicht sogar Nachrichten, wie z.B. Bestellungen per Internet mit einem falschen Namen versandt werden?
Zum einen versucht man diesen Gefahren durch Kodierung der Nachrichten mittels Verschlüsselungsverfahren (s. *Kapitel 2*) zu begegnen. Abgehörte Nachrichten, die nicht zu verstehen sind, da man sie nicht entschlüsseln kann, stellen für einen Angreifer keinen besonderen Nutzen dar. An Nachrichten angefügte Zeitstempel sollen dem Empfänger den Zeitpunkt des Absendens der Nachrichten signalisieren, damit jederzeit ersichtlich wird, ob ein und dieselbe Nachricht evtl.

[1] Daten nennt man *digital*, wenn sie in Form von diskreten Zahlenwerten vorliegen. Die Digitalisierung von Daten ist die notwendige Voraussetzung für die Datenverarbeitung im Computer, da dieser nur mit konkreten Zahlenwerten arbeiten kann.

[2] WWW (Abkürzung für **W**orld **W**ide **W**eb = Weltweites Spinnennetz: ist ein multimediales Informationssystem im Internet; Mit sog. Web-Browsern können sog. Hypertextdokumente mit Text, Symbolen und Bildern auf dem Bildschirm angezeigt werden, deren Quellen irgendwo auf dem Erdball liegen.

unrechtmäßig nochmals verschickt wurde. Dies kann z.B. dann fatale Folgen haben, wenn es sich bei einer solchen Nachricht um einen Zugangscode handelt, der beim Empfang eine wichtige Aktion auslöst, wie z.B. das Zünden einer Rakete. Erhält man eine dementsprechende Nachricht wiederholt mit dem identischen Zeitstempel, so kann der Empfänger davon ausgehen, daß, salopp gesagt: „etwas nicht stimmt".
Wie kann aber nun ein Empfänger erkennen, von wem eine Nachricht stammt? Für sogenannte „Hacker" ist es ein leichtes, z.B. den Absender einer E-Mail zu manipulieren. Elektronische Kaufhäuser vertrauen meist auf die Ehrlichkeit ihrer Kunden bzw. legen Kundenkarteien an, bei denen sich ein Kunde vor der ersten Bestellung registrieren lassen muß, bevor er den Vorteil des Online-Bestellens nutzen kann.
Wünschenswert wäre jedoch ein Verfahren, welches bei einer empfangenen Nachricht bzw. einem Dokument erkennen läßt, von wem es stammt und ob es während der Übertragung verändert wurde. Eine Lösung hierzu sind sogenannte Digitale Signaturen. Da die rechtliche Grundlage für die Verwendung Digitaler Signaturen mit dem Signaturgesetz vom August 1997 geschaffen wurde, kann man davon ausgehen, daß in Zukunft neben der handgeschriebenen Unterschrift, auch die Digitale Signatur zunehmend Verwendung finden wird.

Eine Digitale Signatur stellt zu einer handgeschriebenen Unterschrift das „digitale Gegenstück" dar. Mittels eines Signaturverfahrens wird von Daten, die in elektronischer Form gespeichert sind, eine Kennung erzeugt, die dem Empfänger der Nachricht den Absender des Dokuments anzeigt und es ihm ermöglicht, die Unverfälschtheit der Nachricht zu überprüfen.
In **Kapitel 1** wird zunächst ein Einblick in die Thematik Digitaler Signaturen gegeben, worauf
Kapitel 2 eine kurze Einführung zu Begriffen und Verfahren der Kryptographie gibt, um ein besseres Verständnis für die in
Kapitel 3 geschilderte Funktionsweise Digitaler Signaturverfahren zu ermöglichen.
In **Kapitel 4** wird die Effizienz Digitaler Signaturverfahren in Abhängigkeit von Hashfunktionen erörtert.
In **Kapitel 5** werden die Sicherheitsaspekte Digitaler Signaturen untersucht.
Kapitel 6 gibt Einblick in die möglichen Zertifizierungsinfrastrukturen für öffentliche Schlüssel, die bei Digitalen Signaturverfahren benötigt werden und
Kapitel 7 stellt konkrete Algorithmen vor, die bei der Signaturerstellung Verwendung finden.
Kapitel 8 gibt letztendlich noch Aufschluß über die rechtlichen Grundlagen Digitaler Signaturen und erläutert den praktischen Einsatz derselben im täglichen Leben, worauf
Kapitel 9 die aktuelle Verläßlichkeit Digitaler Signaturen erörtert.

Danksagung

Mein besonderer Dank gilt meinem Betreuer PD Dr. Johannes Köbler, der mir jederzeit mit seinem Rat und seiner konstruktiven Kritik zur Seite stand.

Herrn Prof. Dr. Franz Schweiggert danke ich für die Übernahme der Zweitkorrektur dieser Arbeit.

Zudem möchte ich mich an dieser Stelle bei meinen Eltern für ihr Verständnis und ihre Fürsorge bedanken und v.a. dafür, daß sie mir dieses Studium ermöglicht haben.

Ein Dankeschön gilt auch all meinen Freunden, die mich während dieser Zeit unterstützt haben, insbesondere Sonja Wedel und Gisela Bergmaier für das Korrekturlesen der Diplomarbeit.

Teil I

Theoretische Betrachtungen

Kapitel 1

Einführung in die Thematik Digitaler Signaturen

Seit jeher ist es dem Menschen ein Bedürfnis, eigen Hervorgebrachtes als das eigene Werk zu kennzeichnen. Sei es durch in den Stein einer Statue gemeiselte Initialen oder durch eine, mit einem Wachssiegel versehene Pergamentrolle. Im Laufe der Zeit entwickelte sich die eigene Unterschrift zum Maß aller Dinge: es galt nicht mehr nur der Handschlag, der einen Vertrag besiegelte, sondern der mit Feder und Tinte unterschriebene Vertrag.
Was macht eine Unterschrift so wertvoll?

1.1 Grundlegende Eigenschaften von Signaturen

Authentizität: Eine Unterschrift ist authentisch. Sie überzeugt den Empfänger des Dokuments davon, daß der Unterzeichner das Dokument willentlich unterschrieben hat.

Fälschungssicherheit: Eine Unterschrift ist fälschungssicher. Sie beweist, daß der Unterzeichner und *kein anderer* das Dokument unterschrieben hat.

Einmaligkeit (Einmalige Verwendung): Die geleistete Unterschrift ist an das Dokument gebunden. Sie kann als solches nicht wiederverwendet und in kein anderes Dokument übertragen werden.

Integrität (Unveränderbarkeit): Der Inhalt eines einmal unterschriebenen Dokuments kann nachträglich nicht mehr geändert werden.

Verbindlichkeit: Ein Urheber eines Dokuments sollte später nicht leugnen können, daß er ein Dokument unterschrieben hat.

Heutzutage werden Daten nicht mehr nur in Form von Akten archiviert, sondern sie werden in zunehmenden Maße auch digital gespeichert. Dabei stellt sich die Frage, ob es auch auf digitaler Ebene eine Möglichkeit gibt, Daten auf irgendeine Art und Weise zu unterschreiben, um die Herkunft derselben kenntlich machen zu können? Diese Möglichkeit besteht durchaus: Mittels sog. „Digitaler Signaturverfahren" gelingt es, digitale Unterschriften zu erzeugen. Die Frage, die sich hierbei allerdings stellt ist, ob diese Signaturen die genannten Eigenschaften von Handsignaturen aufweisen können. Bevor dies genauer untersucht wird, sollte zunächst der Begriff „Digitale Signatur" erläutert werden.

Was sind Digitale Signaturen?

Digitale Signaturen bzw. Digitale Signaturverfahren stellen Methoden dar, die es ermöglichen, Daten, die in elektronischer Form gespeichert sind, mit technischen Mitteln zu unterschreiben. Sie dienen außerdem dazu, einem Dokument, das z.B. mit Hilfe eines Softwareprogramms erstellt wurde, einen eindeutigen Erzeuger zuzuordnen. Ein sogenannter Signaturalgorithmus berechnet aus dem zu signierenden Dokument einen speziellen Zahlencode, die sogenannte Digitale Signatur. Besteht der Verdacht, daß ein Dokument gefälscht wurde bzw. der Erzeuger eines Dokuments dieses verleugnen will, so ist man (z.B. der Empfänger) dank der Digitalen Signatur und einem sogenannten Verifikationsalgorithmus in der Lage, die Sachlage aufzuklären.
Der Ablauf eines Digitalen Signaturverfahrens wird in *Kapitel 3* ausführlich geschildert. Nun zurück zu der Frage, ob Digitale Signaturen den Eigenschaften von Handsignaturen entsprechen können, um somit als „digitale Variante" in Betracht gezogen werden zu können. Dazu müssen sie besagte Anforderungen erfüllen, die im folgenden eingehender beschrieben werden.

1.2 Handsignaturen und Digitale Signaturen im Vergleich

Keine Unterschrift ist wie die eines anderen. Jeder Mensch besitzt charakteristische Schriftzüge, anhand derer Graphologen den Urheber einer Unterschrift eindeutig indentifizieren können. Diese Eigenschaft nennt man *Authentizität* (*griech., lat.:* Echtheit, Glaubwürdigkeit). Ein Dokument, welches durch eine handgeschriebene Unterschrift „besiegelt" wurde, kann somit jederzeit dem Unterzeichner eindeutig zugeordnet werden.

Für Digitale Signaturen bedeutet dies, daß es dem Empfänger (dem Verifizierer) eines digital signierten Dokuments möglich sein sollte, die Herkunft dieses Dokuments festzustellen. Der Unterzeichner sollte sich nicht als jemand ausgeben können, der er nicht ist. Wie kann sich der Verifizierer eines digital unterschriebenen Dokuments sicher sein, daß die Unterschrift auch von dem angegebenen Unterzeichner stammt, d.h. daß dieser das Dokument auch willentlich unterschrieben hat?
Um dies gewährleisten zu können wird zum Generieren einer Digitalen Signatur ein Geheimnis notwendig, welches nur der Unterzeichnende selbst kennt. Dies kann z.B. darin bestehen, daß der Signierer in den Signaturalgorithmus eine Geheiminformation (z.B. in Form eines geheimen Schüssels) einbaut, wie es unter *Kapitel 3* beschrieben wird.
Erhält man nun ein digital signiertes Dokument, so kann man davon ausgehen, daß die Signatur unter Mitwirkung des Erzeugers durch Verwendung der Geheiminformation entstanden ist und kann dies mittels des Verifikationsvorgangs (ohne Kenntnis der Geheiminformation) überprüfen. Dies bedeutet, daß auch bei Digitalen Signaturen die geforderte Authentizität erfüllt werden kann.

Eng verbunden mit der Authentizität einer Unterschrift ist deren *Fälschungssicherheit*. In der Regel ist es sehr schwer, außer vielleicht für geübte Fälscher, die Handsignatur eines anderen so nachzuahmen, daß es andere nicht erkennen können. Eine Unterschrift ermöglicht es einem festzustellen, ob der genannte Unterzeichner das Dokument unterschrieben hat oder ob man Zweifel an dem Ursprung der Signatur hegen muß.
Wie verhält es sich mit der Fälschungssicherheit bei Digitalen Signaturen? Bei Digitalen Signaturen kann man Fälschungen nicht einfach nur mit „bloßem Auge" erkennen: Ein Zahlencode aus Nullen und Einsen weist nun eben kein einmaliges Schriftbild auf. Um einer Digitalen Signatur ihren eigenen, charakteristischen „Schriftzug" zuzuweisen, verwendet man meist, wie bereits im vorigen Absatz erwähnt, eine Geheiminformation des Unterzeichnenden. Dadurch erhält jede digitale Unterschrift ihre „eigene Note". Solange die Geheiminformation auch wirklich geheim bleibt, ist die Digitale Signatur auch fälschungssicher in dem Sinne, daß der Verifikationsvorgang eventuelle Veränderungen der Signatur bzw. „falsche Signaturen" aufdecken kann.

Anmerkung:
Ein Unterzeichner sollte sich stets darüber im klaren sein, daß er für die sichere Verwahrung seiner Geheiminformation verantwortlich ist. Ein Verlust dieser Geheiminformation kann dazu führen, daß nicht-autorisierte Personen diese dazu verwenden, Dokumente anstelle des eigentlichen Besitzers zu unterschreiben. In diesem Falle hat der Eigentümer dieses Schlüssels jedoch keine Beweismöglichkeit, daß nicht er, sondern jemand anderes die Digitale Signatur geleistet hat. Deshalb sollte man die Geheimhaltung dieser Information sehr ernst nehmen!

Eine weiterere wesentliche Eigenschaft von Handsignaturen ist ihre *Einmaligkeit*. Dies bedeutet, daß eine Unterschrift nicht unerkannt „einfach" wiederverwendbar ist, sondern einen festen Bestandteil eines Dokuments darstellt. Ein kopiertes Dokument mit „kopierter" Unterschrift, zum Beispiel, besitzt keinerlei Gültigkeit, außer es wurde zusätzlich von einer amtlichen Stelle beglaubigt.
Während es bei handgeschriebenen Signaturen beinahe unmöglich ist, eine Unterschrift wieder von einem Dokument zu entfernen und auf andere Dokumente zu übertragen, ohne daß dies auffallen würde, ist es eine Tatsache, daß das Kopieren von Dokumenten mit Hilfe des Computers und entsprechender Software ein leichtes ist. Würde man eine Digitale Signatur, wie eine „Handsignatur" einfach ans Ende eines Dokuments anfügen, so könnte man diese innerhalb von Minuten z.B. mittels der Funktionen: „Kopieren und Wiedereinfügen" in großen Mengen digital vervielfältigen und auch leicht unter andere Dokumente setzen. Um diesen Mißbrauch digitaler Unterschriften zu verhindern, werden Signaturen immer in Abhängigkeit des zu signierenden Dokuments erstellt. Eine Digitale Signatur eines Erzeugers sieht durch die logische Anbindung an das zu unterzeichnende Dokument tatsächlich niemals gleich aus, läßt aber trotzdem durch den Verifikationsalgorithmus eindeutig Rückschlüsse auf die Identität des Signierenden zu.
Durch den Verifikationsalgorithmus kann man folglich erkennen, ob eine Digitale Signatur zu dem übermittelten Dokument erstellt wurde und somit die Einmaligkeit dieser Unterschriften feststellen.

Diese positiven Eigenschaften der Handsignaturen sind natürlich nur dann garantiert, wenn ein Dokument nach dem Leisten der Unterschrift nicht mehr unbemerkt verändert werden kann. Schließlich möchte jeder nur für das zur Rechenschaft gezogen werden können, was er auch tatsächlich unterschrieben hat. Die *Unveränderbarkeit* bzw. *Datenintegrität* des unterschriebenen Dokuments muß daher gewährleistet sein und ist es bei handschriftlichen Dokumenten im allgemeinen dann, wenn sie nicht nur mit Bleistift, sondern z.B. mit Tinte oder Kugelschreiber verfaßt wurden.
Auch bei einem digital signierten Dokument sollte der „Empfänger" feststellen können, ob dieses bei der Übermittlung durch einen Widersacher verändert wurde. Dies bedeutet, Integrität zielt darauf ab, unautorisierte Veränderungen, wie z.B. Substitution, Einfügen und Löschen von Daten erkennen bzw. verhindern zu können.
Diesen Anforderungen kann man zum Teil mittels physikalischer (datenunabhängiger), nicht-kryptographischer Schutzmethoden gerecht werden, wie z.B. durch Zugangssperren, die dazu dienen, nur autorisiertem Personal Zugang zu gewissen Daten zu gewähren. Meist wird jedoch zum Erhalt von Datenintegriät auf mathematische (datenabhängige) Algorithmen zurückgegriffen. Bei Digitalen Signaturen wird zum einen durch die Einbeziehung des Dokuments in die Signaturbildung versucht, Veränderungen an signierten Dokumenten sichtbar zu machen.

Zum anderen wird durch die Verwendung von Hashfunktionen eine weitere Methode angewandt, um die Übermittlung von signierten Dokumenten sicherer (d.h. ohne Veränderungen) und effizienter gestalten zu können. Wie dies im einzelnen durchgeführt wird, wird in *Kapitel 4* eingehend erläutert. Insgesamt gilt jedoch, daß auch die Datenintegrität bei Digitalen Signaturen eingehalten werden kann.

Ein weiterer wichtiger Gesichtspunkt für die Bewertung Digitaler Signaturen stellt die *Verbindlichkeit* dar. Dies bedeutet, daß man eine Unterschrift, die man einmal „gegeben" hat, im nachhinein nicht verleugnen kann.
Bei „Handsignaturen" ist es einleuchtend, daß man eine Unterschrift nicht zurückweisen kann. Die Unterschrift liegt physisch vor und im Zweifelsfall kann spätestens ein Graphologe feststellen, ob eine Unterschrift ein Original oder eine Fälschung ist.
Bei Digitalen Signaturen ist dies nicht so einfach, da man als Unterzeichner prinzipiell behaupten kann, jemand sei auf irgendeine Weise in den Besitz der Geheiminformation gelangt und habe deshalb die Digitale Signatur fälschen können. Prinzipiell liegt jedoch die Verantwortung für die Geheiminformation beim Unterzeichnenden selbst, d.h. die sichere Verwahrung derselben ist ein absolutes Muß!

Zusammenfassend kann man feststellen, daß Digitale Signaturen dieselben Eigenschaften wie Handsignaturen aufweisen können, wenn dies auch an bestimmte Bedingungen geknüpft ist, wie z.B. dem Verwenden einer Geheiminformation oder dem „logischen Verknüpfen" der Signatur mit dem Inhalt des zu signierenden Dokuments.
Tatsache ist, daß es nicht für alle Anforderungen gleichzeitig eine Lösung gibt, sondern, daß es für das Erfüllen der gewünschten Eigenschaften *Authentizität, Fälschungssicherheit, Einmaligkeit* und damit einhergehenden Anforderungen wie *Datenintegrität* und *Verbindlichkeit* häufig unterschiedlicher Vorgehensweisen bedarf. Zudem sollte man bei all diesen Anforderungen die Gewährleistung von *Vertraulichkeit* nicht außer Acht lassen. Vertraulichkeit zielt darauf ab, den Inhalt von Dokumenten bzw. Informationen ausschließlich autorisierten Personen zukommen zu lassen. In *Kapitel 3* und *4* wird ersichtlich, wie Digitale Signaturen und kryptographische Verfahren bzw. Digitale Signaturen und Hashfunktionen zusammenspielen, um die genannten Anforderungen erfüllen zu können.
Im folgenden Abschnitt wird nun eine Übersicht über den möglichen Einsatz Digitaler Signaturen gegeben, da sie neben den altbekannten Nutzungsmöglichkeiten von Handsignaturen auch neue Anwendungsbereiche eröffnen.

1.3 Einsatz Digitaler Signaturen

Viele Regelungen des Zusammenlebens in einer Gesellschaft gründen auf Papier: Verträge, Anträge, Formulare, Klagen, Urteile, sowie Zeugnisse und Ausweise werden „schwarz auf weiß" festgehalten und anschließend signiert. Hierbei stößt man immer häufiger auf Probleme bei der Archivierung von Dokumenten: Die Frage ist, wohin mit der heutigen Dokumentenflut? Eine naheliegende Lösung zur Platzersparnis stellt mittlerweile die Möglichkeit dar, Dokumente elektronisch zu speichern. Digitale Signaturen wären hierbei für derartige Dokumente von großem Vorteil, um diese eindeutig einem „Urheber" zuordnen zu können. Aufgrund ihrer Eigenschaften können Digitale Signaturen prinzipiell überall dort eingesetzt werden, wo bisher „Handsignaturen" Verwendung finden. Zudem eröffnen sich aber auch viele neue Anwendungsgebiete: Mit Digitalen Signaturen können Dateien beliebigen Inhalts, d.h. auch Graphiken und Audio-/Video-Sequenzen digital signiert werden, v.a. letzteres ist mit Handsignaturen undenkbar. Auch für den Einsatz bei Anwendungen im „Client-/Server-Bereich" werden Digitale Signaturen zudem immer wichtiger: Sei es bei der Datenarchivierung in Krankenhäusern oder in Ämtern zum vertraulichen Umgang mit personenbezogenen Daten, bei Anwendungssoftware mit integrierter Signierfunktion, oder auch bei Vertragsübermittlungen, wie z.B. beim Einkaufen in „elektronischen Kaufhäusern" über das Internet. Speziell durch den Einsatz Digitaler Signaturen im Bereich des „Homebanking" und beim elektronischen Datenaustausch von Behörden kann durch die kürzeren Übermittlungszeiten extrem viel Zeit und damit auch Geld gespart werden.
Die Verwendung Digitaler Signaturen beim Datentransport in offenen Telekommunikationsnetzen ermöglicht die Überprüfung, von wem ein Dokument stammt und ob es bei der Übermittlung verfälscht wurde. Aufgrund all dieser postiven Eigenschaften bildet die Digitale Signatur eine Basis für einen sicheren Austausch von Daten in der Computerkommunikation und über das Internet, was früher oder später dazu führen wird, daß Digitale Signaturen auch im „täglichen Leben" unverzichtbar werden.

Bevor nun auf die Funktionsweise Digitaler Signaturen und Sicherheitsvorkehrungen im einzelnen eingegangen werden kann, werden grundlegende kryptographische Begriffe und Verfahren vorgestellt, da diese den Einstieg in das Gebiet Digitaler Signaturverfahren um einiges erleichtern.

Kapitel 2

Eine kleine Einführung in die Kryptographie

Wie unter *Kapitel 1* bereits angedeutet wurde, ist es für Digitale Signaturen von besonderer Bedeutung, daß die *Vertraulichkeit* bei der Übermittlung von Daten gewahrt bleibt, d.h. es sollte garantiert sein, daß die Daten nur zu autorisierten Personen gelangen bzw. von diesen empfangen werden. Dies kann durch den Einsatz von Kryptosystemen (s. 2.2) erreicht werden. Zudem spielen Kryptosysteme eine entscheidende Rolle bei der Generierung Digitaler Signaturen (s. *Kapitel 3*), da diese von Kryptosystemen hergeleitet werden können. In diesem Kapitel findet nun eine kleine Einführung in die Begriffswelt der Kryptographie und der Kryptosysteme statt, um im Anschluß darauf Bezug nehmen zu können.

2.1 Grundlegende Begriffe

Was ist eigentlich Kryptographie? Im „klassischen Sinn" gilt die „Kryptographie als eine „Disziplin", die sich mit der Entwicklung von Verschlüsselungsverfahren (s. 2.2) zum Schutz (geheimer) Daten vor unbefugten Zugriffen befaßt." [BI93]. Mittlerweile umfaßt die Kryptographie nicht mehr „nur" Methoden zur Verschlüsselung (*Chiffrierung*) und Entschlüsselung (*Dechiffrierung*) von Daten, sondern sie dient auch genauesten wissenschaftlichen Analysen von Kryptosystemen auf ihre Sicherheit. Zudem sind viele Bereiche, wie z.B. Steganographie[1], Digitale Signatur- und Identifizierungsverfahren u.v.m. hinzugekommen, die das Ziel haben, Angriffe auf die Sicherheit von Daten durch entsprechende Verfahren zu vereiteln bzw. gewisse Informationen gegenüber unbefugten Personen geheimzuhalten.
Speziell der Bereich der **Kryptanalyse** versucht durch die Analyse von *Kryptoverfahren*, deren Stärken und v.a. Schwächen herauszufinden, um sie im Hinblick

[1] *Steganographie* ist die Lehre der Geheimhaltung von Information, durch das Verbergen ihrer Existenz.

auf ihre Sicherheit bewerten zu können.
Die **Kryptanalysis** hingegen versucht, mit unterschiedlichsten Mitteln, verschlüsselten Text (*Chiffretext*) aufzubrechen, mit dem Ziel, die ursprüngliche Nachricht zumindest teilweise rekonstruieren zu können.
Die **Kryptologie** stellt insgesamt die Wissenschaft der Geheimhaltung von Informationen durch Verschlüsselung von Daten dar. Sie beschreibt den mathematischen Teil, der sowohl Kryptographie als auch Kryptanalyse umfaßt.
Hierbei sind die wichtigsten *Ziele* kryptographischer Verfahren die Wahrung von *Integrität* und von *Vertraulichkeit* (s. Def. *Kapitel 1*) von Nachrichten bzw. Daten.

2.2 Kryptosysteme

Definition: *Kryptosystem*

Ein *Kryptosystem* (s. Abb. 2.1) setzt sich aus einem *Verschlüsselungs-* und *Entschlüsselungsalgorithmus* zusammen. Dabei überführt es einen Klartext (eine beliebige Folge von Zeichen) aus einem bestimmten Zeichenvorrat, dem *Klartextalphabet* unter Verwendung eines *Schlüssels* in einen zugehörigen *Kryptotext*, der wiederum aus einem Zeichensatz, dem *Kryptotextalphabet* gebildet wird. (Klartext- und Kryptotextalphabet können durchaus denselben Zeichensatz darstellen). Der Bereich, aus dem die Ver- und Entschlüsselungsschlüssel gewählt werden, nennt man *Schlüsselraum.*
Welche Möglichkeiten gibt es nun, Verschlüsselungen durchzuführen?
Prinzipiell unterscheidet man hierbei zwei Verfahren:
Die *symmetrischen* und die *asymmetrischen* Kryptosysteme.

2.2.1 Symmetrische Kryptosysteme

In einem symmetrischen Kryptosystem verwenden Sender und Empfänger den gleichen Schlüssel $k \in K$. Die Ver- und Entschlüsselung findet somit unter Verwendung desselben Schlüssels statt.
In Abb. 2.1 wäre dann $k = \overline{k} = k_{AB}$ der gemeinsame Schlüssel von Alice und Bob. Dieser muß auf „abhörsicherem Wege", z.B. in Form eines gesicherten Kanals[2] vor Beginn der Kommunikation zu den Kommunikationsteilnehmern gelangen, damit geheime Kommunikation stattfinden kann.
Zu den symmetrischen Kryptosystemen gehören viele klassische Verschlüsselungsmethoden wie *monographische, polygraphische Substitutionen* und *Transpositionschiffren.* Diese zeichnen sich dadurch aus, daß ein oder mehrere Zeichen nach be-

[2] *Gesicherter* Kanal bedeutet, daß Nachrichten oder Daten, die über diesen Kanal übertragen werden, weder von Unbefugten eingesehen, noch manipuliert werden können. Dies bedeutet zudem, daß die Vertraulichkeit der Daten gewahrt ist, d.h. der Empfänger kann sich sicher sein, daß eine empfangene Nachricht vom angegebenen Sender stammt.

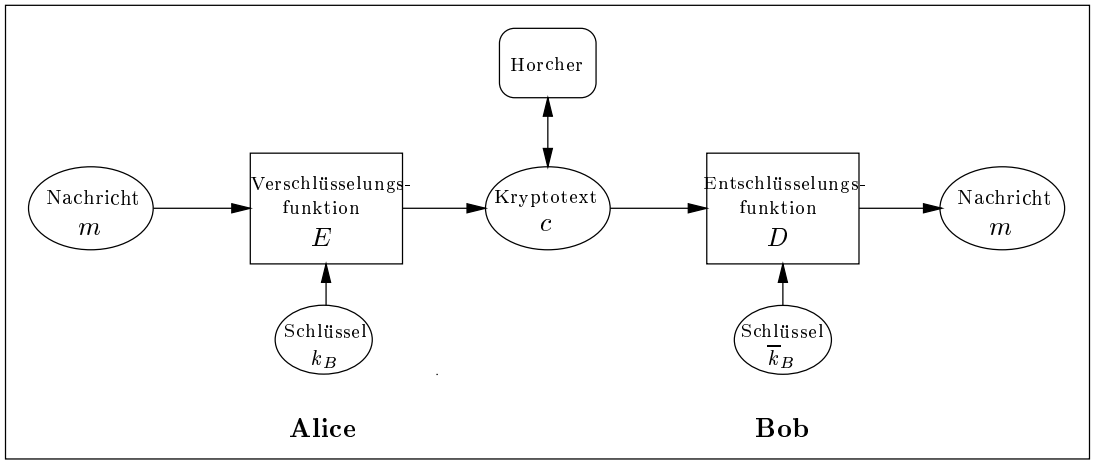


Abbildung 2.1: Die Abbildung zeigt die schematische Darstellung eines Kryptosystems

m	Nachricht, $m \in M$; M: Nachrichtenraum
c	Kryptotext,$c \in C$; C: Kryptotextraum
E	Verschlüsselungsfunktion: $E : X \times K \to C$
D	Entschlüsselungsfunktion: $D : C \times \overline{K} \to X$
$k, \overline{k}$	(Chiffrier-, Dechiffrier-)Schlüssel, $k \in K$, $\overline{k} \in \overline{K}$
$K, \overline{K}$	Schlüsselräume (Chiffrierung und Dechiffrierung)

Alice:	führt die Verschlüsselung einer Nachricht m durch.
Bob:	erhält verschlüsselte Nachrichten c und entschlüsselt diese.
k_B:	ist der (öffentliche) Chiffrierschlüssel von Bob.
$\overline{k}_B$:	ist der (geheime) Dechiffrierschlüssel von Bob.
Horcher:	ist ein Angreifer (Kryptanalytiker), der übermittelte Nachrichten unberechtigter Weise abhört und evtl. versucht, das Kryptosystem zu brechen .

stimmten vorgegebenen Regeln durch andere Zeichen substituiert bzw. in andere Zeichen transformiert werden, um den Text für „Nicht-Eingeweihte" unleserlich zu gestalten. Da jedoch bei diesen Verschlüsselungen, die Buchstabenhäufigkeiten erhalten bleiben, bietet diese Eigenschaft einen guten Angriffspunkt für die Kryptanalyse. Um dies zu verhindern, ging man auf die Verwendung sogenannter Produktchiffren über. Eine *Produktchiffre* erhält man dadurch, daß man sequentiell ein oder mehrerere Verschlüsselungsverfahren auf den Klartext anwendet. Durch die Kombination ein bzw. mehrerer Verfahren sind Produktchiffren meist extrem schwer zu brechen, auch wenn ihre Komponenten, die z.B. einfache Grundfunktionen darstellen, einzeln leicht zu brechen wären. [Kö94]
Beispiele für Produktchiffren sind der *DES-* , der daraus entwickelte *Dreifach-DES-* und der *IDEA-Algorithmus* (s. Anhang A). Diese werden auch häufig in Systemen zur Bildung Digitaler Signaturen verwendet.

Ein wesentlicher *Nachteil* symmetrischer Kryptosysteme stellt jedoch die Tatsache dar, daß zur Ver- und Entschlüsselung ein- und derselbe Schlüssel (ein sog. *symmetrischer Schlüssel*) verwendet wird. Diese müssen vor Beginn der geheimen Kommunikation über einen „gesicherten Kanal" ausgetauscht werden, da die Kommunikationssicherheit von der Geheimhaltung dieser Schlüssel abhängt. Dies stellt zur Verschlüsselung einen zusätzlichen Aufwand dar, der durch die Verwendung sogenannter asymmetrischer Kryptosysteme, auch Public-Key-Systeme genannt, vermieden werden kann, da bei diesen Systemen zum Schlüsselaustausch nur „authentisierte"[3] Kanäle benötigt werden (s.u.).

2.2.2 Asymmetrische Kryptosysteme

Das Prinzip der *asymmetrischen Kryptosysteme* der sog. *Public-Key-Kryptographie* wurde erstmals von W. Diffie und M. Hellman, sowie unabhängig davon von R. Merkle erfunden. Diffie und Hellman stellten das Konzept erstmals auf der National Computer Conference 1976 vor. Der entscheidende Gedanke war, daß Schlüssel paarweise, als *Chiffrier-* und *Dechiffrier- Schlüssel* (sog. *asymmetrische Schlüssel*) auftreten, wobei ein Schlüssel nicht aus dem anderen konstruiert werden kann. Einer der beiden Schlüssel ist geheim, der andere öffentlich bekannt und muß deshalb nicht über gesicherte Kanäle übermittelt werden, sondern authentisierte[3] Übertragungskanäle reichen aus, damit die Integrität der Daten gewahrt bleiben kann.
Public-Key-Verfahren, wie z.B. RSA oder ElGamal, werden häufig zur Generierung Digitaler Signaturen verwendet und unter *Kapitel 7* eingehender vorgestellt.

[3]Ein *authentisierter* Kanal besitzt die gleichen Eigenschaften eines gesicherten Kanals, jedoch kann die Vertraulichkeit der übermittelten Nachrichten nicht garantiert werden.

Im folgendem wird nun erläutert, wie man Ver- und Entschlüsselung bei dieser Art von Kryptosystemen durchführt, um daraufhin in *Kapitel 3* den Bezug zu kryptographischen Signaturverfahren einleuchtender herstellen zu können.
Wie geht man nun bei der Ver- und Entschlüsselung bei diesen Kryptosystemen vor? Angenommen, Alice möchte mittels eines Public-Key-Kryptosystems eine Nachricht m (bzw. ein Dokument m) verschlüsseln und an Bob übermitteln, der es anschließend wieder entschlüsselt. Welche Schritte müssen dazu durchgeführt werden?

- **Ablauf des Verschlüsselns**

  1 Alice erhält von Bob den öffentlichen Schlüssel k_B bzw. kann sich auf irgendeine andere Weise diesen Schlüssel beschaffen.

  2 Alice verschlüsselt die Nachricht m mit dem Schlüssel k_B, indem sie die Verschlüsselungsfunktion E auf m folgendermaßen anwendet:
  $E(m, k_B) = c$ und berechnet somit den zu m zugehörigen Kryptotext c.

  3 Alice übermittelt die verschlüsselte Nachricht c an Bob.

  Welche Berechnungen muß Bob durchführen, um aus dem Kryptotext c die ursprüngliche Nachricht zu erhalten?

- **Ablauf des Entschlüsselns**

  1 Bob entschlüsselt den Kryptotext c mittels seines geheimen Schlüssels $\overline{k}_B$ und der zu E korrespondierenden Entschlüsselungsfunktion D, indem er folgende Berechnung durchführt:
  $D(c, \overline{k}_B) = m$, da folgendes gilt: $D(c, \overline{k}_B) = D(E(m, k), \overline{k}_B) = m$.

  2 Dadurch erhält Bob die ursprünglich verschlüsselte Nachricht m.

Die Übermittlung einer verschlüsselten Nachricht von Bob an Alice erfolgt analog dem soeben Beschriebenen, wobei Bob zum Verschlüsseln den öffentlichen Schlüssel von Alice verwenden muß, um es nur für sie lesbar zu machen.

Anmerkung:
Man spricht bei Public-Key-Systemen von *asymmetrischen Kryptosystemen*, da nur Bob und niemand sonst die verschlüsselten Nachrichten (Dokumente), die an ihn gerichtet sind, entschlüsseln kann. Bei symmetrischen Kryptosystemen verfügen die Kommunikationspartner, wie bereits erwähnt, für den geheimen Datenaustausch über einen gemeinsamen, geheimen Schlüssel, wodurch die Ver- und Entschlüsselung bei beiden identisch (bzw. symmetrisch) abläuft.

Damit die beschriebenen Kryptosysteme auch wirklich „einwandfrei" funktionieren können, sollten jedoch gewisse Anforderungen erfüllt sein.

2.3 Anforderungen an Kryptosysteme

Ziele von Kryptosystemen sind, wie bereits erwähnt, v.a. die Gewährleistung der Datenintegrität der übermittelten Nachrichten (bzw. Dokumente) und die Erfüllung von Vertraulichkeit. Um dies zu erreichen, sollte ein Kryptosystem besondere Eigenschaften aufweisen und gewisse Gütekriterien und Anforderungen erfüllen, um Sicherheit bieten zu können.

- **Notwendige Bedingungen für Kryptosysteme:**
  - Die Schlüssel $k \in K$ und $\overline{k} \in \overline{K}$ sollten so gewählt, daß gilt:
    $\forall m \in M$: $D(E(m,k),\overline{k}) = m$. (*Korrektheitseigenschaft*)
  - Insbesondere muß für jedes $k \in K$ die Funktion $m \to E(m,k)$ injektiv sein.
    Anm.:
    Dies wird gefordert, damit jeder Kryptotext eindeutig der ursprünglichen Nachricht zugeordnet werden kann.

- Zudem sollten folgende **Gütekriterien für Kryptosysteme** erfüllt sein:
  - Die Chiffrierfunktion E und die Dechiffrierfunktion D sollten einfach zu berechnen sein, um das Erstellen des Kryptotextes in einem annehmbaren, zeitlichen Rahmen halten zu können.
    (*Berechnungseffizienz*)
  - Ohne den (geheimen) Dechiffrierschlüssel $\overline{k}$ zu kennen, sollte es unmöglich[4] sein, $D(c,\overline{k})$ zu berechnen, da die Sicherheit des Systems wesentlich davon abhängt. (*Fälschungssicherheit*)

- **Komplexitätstheoretische Sicherheit**
  Eine wichtige Erkenntnis bei Public-Key-Kryptosystemen ist, daß diese Systeme keine informationstheoretische, sondern höchstens komplexitätstheoretische Sicherheit aufweisen können.
  Informationstheoretische Sicherheit:
  Man spricht bei Kryptosystemen dann von informationstheoretischer bzw. uneingeschränkter Sicherheit, wenn es einem Gegner auch nicht mit uneingeschränkten Rechenressourcen möglich ist, das System zu brechen.
  Komplexitätstheoretische Sicherheit:
  Ist es für einen Gegner nicht möglich, das Kryptosystem mit einem für ihn lohnenswerten Aufwand zu brechen, so spricht man von einem komplexitätstheoretisch- bzw. berechnungssicheren System.

[4] unmöglich heißt in diesem Falle, nicht mit den heute und auf absehbare Zeit zur Verfügung stehenden Mitteln (Computer, Rechenressourcen, etc.)

Warum sind Public-Key-Systeme nicht informationstheoretisch sicher?
Bei einem Public-Key-Verfahren ist, wie man bereits feststellen konnte, die Chiffrierfunktion, sowie der Chiffrierschlüssel öffentlich bekannt. Ein Widersacher kann z.B. einen *Brute-Force-Angriff* (engl.: rohe Gewalt) folgendermaßen durchführen:
Bei einem derartigen Angriff (auch Vorwärtssucheangriff) führt der Gegner eines Kryptosystems eine (vollständige) Klartextsuche durch. Dies bedeutet, er verschlüsselt alle in Frage kommenden Klartexte mit der bekannten Chiffrierfunktion und bekanntem Chiffrierschlüssel, bis er den beobachteten (abgehörten) Kryptotext erhält.
Dies bedeutet, daß prinzipiell die Möglichkeit besteht, zu einem gegebenen Kryptotext auch ohne Kenntnis des geheimen Dechiffrierschlüssels, den zugehörigen Klartext zu ermitteln, d.h. die gewünschte informationstheoretische Sicherheit ist hierbei verletzt.
Damit ein Public-Key-Kryptosystem trotzdem sicher ist, muß es zumindest besagte komplexitätstheoretische Sicherheit aufweisen, d.h. z.B. durch Verwendung entsprechend langer Schlüssel Berechnungen (wie z.B. beim Brute-Force-Angriff) so aufwendig zu machen, daß es den zeitlichen Rahmen bzw. berechnungstechnischen Rahmen „sprengt". Wie lange sollten nun symmetrische bzw. asymmetrische Schlüssel mindestens sein, um derartige Angriffe vereiteln zu können?

- **Bedeutung der Schlüssellänge für Kryptosysteme**
  Die Schlüssellänge der in Kryptosystemen verwendeten Schlüssel trägt wesentlich zur Sicherheit eines Kryptosystems bei. Durch entsprechend lange Schlüssel, kann man z.B. Brute-Force-Angriffe vereiteln, indem die besagten Berechnungen dadurch so aufwendig werden können, daß sie nicht in absehbarer Zeit (mit den zur Zeit zur Verfügung stehenden technischen Mitteln) durchgeführt werden können. Um die Schlüsselänge(n) zu ermitteln, die ein Mindestmaß an Sicherheit gewährleisten, kann man folgende Abschätzungen durchführen.
  Bei *symmetrischen Verschlüsselungsverfahren* sind bei einer Schlüssellänge von beispielsweise 8 Bit, $2^8 = 256$ mögliche Schlüssel generierbar, d.h. mittels besagter *Brute-Force-Methode* hat man nach spätestens 256 Versuchen (bei bekannter Klartext- / Chiffretext-Sequenz) den richtigen Schlüssel gefunden. Wird hierbei mit einer geschätzten 50%igen Wahrscheinlichkeit, bereits nach der Hälfte aller Versuche der richtige Schlüssel gefunden wird, so müßte man nur 128 bis maximal 256 Schlüssel testen. Diese Tests können in ziemlich kurzer Zeit durchgeführt werden, so daß diese Schlüssellänge als absolut unsicher zu bewerten ist. Erst ab einer Länge von 112 Bit scheint die Gewißheit zu bestehen, daß man auch bei einem Einsatz von 10^{13} Dollar für den dazu benötigten technischen und personellen Aufwand immer noch 10^6 Jahre benötigt, um einen dementsprechend langen Schlüssel nach dieser

Methode finden zu können. Kostenmäßig lassen Militärausgaben momentan laut ([Sch96], S.179) das Brechen von symmetrischen Kryptosystemen bis zu 64 Bit-Schlüsseln zu, was den Einsatz von mindestens 112 Bit langen Schlüsseln untermauert und den Einsatz von 128 Bit langen Schlüsseln dann nahelegt, wenn Schlüssel für eine Lebensdauer von bis zu 30 Jahren vorgesehen sind.
Bei *asymmetrischen Verschlüsselungsverfahren* wird mit öffentlich bekannten Schlüsseln gearbeitet. Um auch hier auf eine sichere Schlüssellänge schließen zu können, muß das Generieren der öffentlichen Schlüssel berücksichtigt werden. Baut z.B. das Entwickeln eines Schlüssels wie beim häufig verwendeten RSA-Verfahren (s. *Kapitel 7*) auf dem *Faktorisierungsproblem* auf, d.h. auf der Tatsache, daß das Multiplizieren zweier großer Primzahlen relativ einfach ist, das Faktorisieren einer aus zwei großen Primzahlen bestehender Zahl zumindest zum jetzigen Kenntnisstand nur in nichtpolynomialer Zeit lösbar ist, so hängt die Sicherheit von der Größe der Faktoren ab. Kennt man die Faktoren, so ist die Sicherheit des Systems nicht mehr gewährleistet. Man darf bei der Bewertung der Schlüssellänge nicht vergessen, daß neue Algorithmen entwickelt werden, um z.B. das Faktorisierungsproblem für immer größere Zahlen in absehbarer Zeit lösen zu können. (Für den diskreten Logarithmus, auf den viele andere Algorithmen beruhen, wie z.B. das ElGamal-System (s. *Kapitel 7*) gilt entsprechend dasselbe.) So benötigt man mittels des sogenannten Zahlkörpersiebverfahrens ([Sch96], S. 189) zur Faktorisierung von 512-Bit langen Schlüsseln bereits weniger als 200 Mips-Jahre (Million-Instructions per Seconds) und erst ab 1024 Bit „noch sichere" $3 * 10^7$ Mips-Jahre.
Insgesamt bedeutet dies, daß man in symmetrischen Kryptosystemen Schlüssel mit einer Mindestlänge von 128 Bit verwenden sollte und in asymmetrischen Kryptosystemen derzeit bei Schlüsseln von 1024 Bit empfehlenswert sind, um (zumindest gegenüber Brute-Force-Angriffen) ausreichend Sicherheit gewährleisten zu können.

Neben der Länge der verwendeten Schlüssel, sollten bei Public-Key-Kryptosystemen zudem die eingesetzten Chiffrier- bzw. Dechiffrierfunktionen ganz bestimmte Eigenschaften aufweisen, um die Sicherheit dieser Verfahren gewährleisten zu können.

- **Verwendete Verschlüsselungsfunktionen bei asymmetrischen Kryptosystemen**
  Aus den genannten Anforderungen bzw. Gütekriterien geht hervor, daß man sich von Verschlüsselungsverfahren einiges erwartet. Zum einen, erhofft man sich, daß es relativ einfach und schnell möglich ist, eine Nachricht (bzw. einen Text) zu verschlüsseln. Zum anderen sollte es jedoch für Kryptanalytiker ohne den Besitz des geheimen Schlüssels so gut wie unmöglich sein, einen erstellten Kryptotext (zumindest nicht in absehbarer Zeit) entschlüsseln zu können. Hierbei spricht man von der *Einwegeigenschaft*, die die sog. Einwegfunktionen erfüllen.

  Definition: *Einwegfunktion*
  Eine injektive Funktion f ist eine *Einwegfunktion*, wenn sie einfach (effizient) berechenbar ist, deren Umkehrfunktion[5] jedoch nicht.
  Dies bedeutet, daß eine Funktion mit der Eigenschaft $f(x) = y$ einfach zu berechnen ist, aber zu gegebenen y, ab einer bestimmten Größe des Problems, x nur noch in nicht-polynomialer Zeit bestimmt werden kann, selbst wenn der Algorithmus zur Berechnung von f und der öffentliche Schlüssel k bekannt sind.

  Zudem sollte es für autorisierte Personen möglich sein, die Entschlüsselung relativ einfach durchführen zu können, d.h. für diese sollte es leicht sein, die Umkehrfunktion zu der verwendeten Verschlüsselungsfunktion zu berechnen. Dies bedeutet aber, daß die bisher geforderte Einwegeigenschaften für die in Public-Key-Systemen verwendeten Algorithmen noch nicht ausreichend sein kann. Die soeben geschilderten *Trapdoor-Eigenschaften* erfüllen die sog. Trapdoor-Funktionen.

  Definition: *Trapdoor-Funktion*
  Eine *Trapdoor-Funktion* f ist eine Einwegfunktion mit der Eigenschaft, daß sie unter Verwendung einer Zusatzinformation, der „Geheimtür" effizient invertierbar[5] wird.
  Es gilt:

  1 f ist einfach zu berechnen.
  2 Bei Kenntnis einer mit f verbunden Information (sog. Geheiminformation) ist $f(x) = y$ bei gegebenem y leicht lösbar. (*Trapdoor-Eigenschaft*)
  3 Ohne Kenntnis der Falltürinformation ist $f(x) = y$ nur schwer lösbar. (*Einwegeigenschaft*)

[5] Eine Funktion f heißt *invertierbar* oder umkehrbar, falls eine *Umkehrfunktion* f^{-1} existiert, so daß gilt: $f(f^{-1}(x)) = x$.

- *Anmerkung:*
  Es konnte bisher noch nicht bewiesen werden, daß derartige Funktionen tatsächlich existieren. Nur unter der Annahme, daß $P \neq NP$, ist die Existenz von Trapdoor-Funktionen nachweisbar, wobei die Bedingung $P \neq NP$ hierzu notwendig ist, aber u.U. *nicht* hinreichend. [Sch92].
  Funktionen, die gute Kandidaten für Trapdoor-Funktionen darstellen, da sie die geforderten Eigenschaften bisher erfüllen, sind zum Beispiel modulare Potenzfunktionen[6] und quadratische Polynomfunktionen[7] modulo dem Produkt n.

Ein *Nachteil* asymmetrischer Kryptosysteme gegenüber symmetrischen tritt bei ihrer *Verarbeitungsgeschwindigkeit* auf: In der Praxis hat sich gezeigt, daß asymmetrische Kryptosysteme wesentlich langsamer (es handelt sich hierbei um 100-fache bis 1000-fache langsamere Werte) Berechnungen durchführen, als symmetrische. Deshalb werden symmetrische und asymmetrische Kryptosysteme häufig kombiniert angewandt: asymmetrische zur Vereinbarung eines symmetrischen Sitzungsschlüssels, symmetrische für den eigentlichen Datenaustausch. Diese sog. *Hybridverfahren* finden auch bei Digitalen Signaturverfahren, wie z.B. bei PGP, welches im *Kapitel 6* vorgestellt wird, häufige Verwendung.

In diesem Kapitel wurde eine kurze Einführung in das Gebiet der Kryptographie und speziell in den Bereich „asymmetrischer Kryptosysteme" gegeben. Hierbei konnte festgestellt werden, daß für asymmetrische Kryptosysteme die Verwendung sog. Trapdoorfunktionen unerläßlich ist, um ausreichend Sicherheit gegenüber Kryptanalytikern gewährleisten zu können. Zudem stellt die verwendete Schlüssellänge aufgrund des Brute-Force-Angriffs einen weiteren, wesentlichen Sicherheitsaspekt dieser Systeme dar. Die Kenntnis dieser Tatsachen ist notwendig, um im folgenden Kapitel, s. *Kapitel 3* aufzeigen zu können, wie sich *kryptographische Digitale Signaturverfahren* von Public-Key-Kryptosystemen herleiten lassen und welche Sicherheitsvorkehrungen bei diesen Schemata zu treffen sind.

[6] Modulare Potenzfunktion: $f(x) = x^e \ mod \ n$, wobei $n = pq$, p,q $prim$ und $ggT(e, \varphi(n)) = 1$; $\varphi(n) = (p-1)(q-1)$

[7] Quadratische Polynomfunktion: $f(x) = a_2x^2 + a_1x + a_0 \ mod \ n$,
wobei $n = pq$, wobei p und q große Primzahlen darstellen.

Kapitel 3

Funktionsweise digitaler Signaturen

Wie bereits erwähnt, zielen Digitale Signaturen v.a. darauf ab, die Authentizität einer Unterschrift überprüfen zu können, um sicher sein zu können, daß das signierte Dokument unverfälscht vorliegt.
In *Kapitel 1* wurde bereits geschildert, daß man hierzu eine „Geheiminformation" benötigt, mit der man einer digitalen Signatur „sein Markenzeichen" aufdrücken kann. Verwendet man als eine derartige Information einen *symmetrischen Schlüssel*, so spricht man von dieser Art von Signaturverfahren als „Message Authentication Codes" (kurz: MAC), wird diese Geheiminformation durch *asymmetrische Schlüssel* repräsentiert, so spricht man von einer Digitalen Signatur. Im folgenden wird nun kurz darauf eingegangen, wodurch sich *MACs* auszeichnen und warum sie in der Praxis meist den Digitalen Signaturverfahren zur Unterschriftengenerierung „weichen" müssen, um im Anschluß daran die Funktionsweise Digitaler Signaturen eingehender zu schildern.

3.1 Message Authentication Codes versus Digitale Signaturen

Was ist ein MAC (Message Authentication Code)?
Ein Message Authentication Code stellt ein kryptographisches Verfahren dar, welches unter Verwendung sog. Hashfunktionen, s. *Kapitel 4* und symmetrischer (geheimer) Schlüssel versucht, die Authentizität von Nachrichten sicherzustellen. Hierbei wird eine Nachricht bzw. ein Dokument mittels einer Hashfunktion und der Eingabe des geheimen Schlüssels so verändert, daß nur der Empfänger, der ebenfalls den geheimen Schlüssel besitzt, dieses wieder entziffern kann. Gelingt ihm dies, so kann er sich sicher sein, daß besagte Nachricht von dem entsprechenden Kommunikationspartner stammt, ansonsten geht er von einer Fälschung

während der Übermittlung aus. Eine detailliertere Beschreibung dieser Verfahren findet sich im Zusammenhang mit der Definition von Hashfunktionen unter *Kapitel 4*.
Ein wesentliches Manko bei dieser Art der Nachrichtenauthentikation besteht darin, daß die Kommunikationspartner nur einander die Authentizität der jeweils empfangenen Nachricht beweisen können, die gewünschte *Verbindlichkeit* gegenüber „Dritten" ist jedoch nicht gewährleistet. Dies bedeutet, möchte man auch jemand anderem als dem direkten Kommunikationspartner beweisen, daß dieser eine Nachricht verfaßt hat, so kann er dies durch die Verwendung von MACs nicht.
Verfahren, die neben Authentizität auch Verbindlichkeit gegenüber Dritten garantieren können, sind besagte *Digitale Signaturverfahren*. Besteht absolutes Vertrauen zwischen den Kommunikationspartnern, so kann auf die aufwendigeren Digitalen Signaturverfahren verzichtet werden, ansonsten empfiehlt es sich auf diese zurückzugreifen, wie es im folgenden beschrieben wird.

3.2 Formale Definition von Digitalen Signaturen

Wie bereits in *Kapitel 1* erwähnt, kann man Digitale Signaturen als das „digitale Gegenstück" zu Handsignaturen bezeichnen. Im folgenden wird nun erläutert, aus welchen Komponenten ein Digitales Signaturverfahren im einzelnen besteht und auf welche Weise man derartige Signaturen generieren kann.

Eine **Digitale Signatur** ist eine Zeichenfolge, welche einem Dokument, das in digitaler Form vorliegt, den Unterzeichner des Dokuments eindeutig zuordnet.
Dies geschieht mittels eines Digitalen Signaturschemas.

Definition: *Digitales Signaturschema*
Ein Digitales Signaturschema besteht aus zwei Funktionen:
Zum einen aus einer **Signaturfunktion** (bzw. einem Signaturalgorithmus) zur Erzeugung digitaler Signaturen, zum anderen aus einer **Verifikationsfunktion** (bzw. einem Verifikationsalgorithmus), zur Überprüfung einer digitalen Unterschrift auf Authentizität.

Das Zusammenspiel dieser beiden Funktionen demonstriert folgendes Beispiel:
Sei $X = (x_1, x_2, x_3)$ die Menge aller Dokumente, die signiert werden können und sei $Z = (z_1, z_2, z_3)$ die Menge aller Unterschriften, die zu diesen Dokumenten durch die Signaturfunktion $S_{\overline{k}}$ generiert wurde. Der Schlüssels $\overline{k}$ stellt die hierbei verwendete Geheiminformation dar.

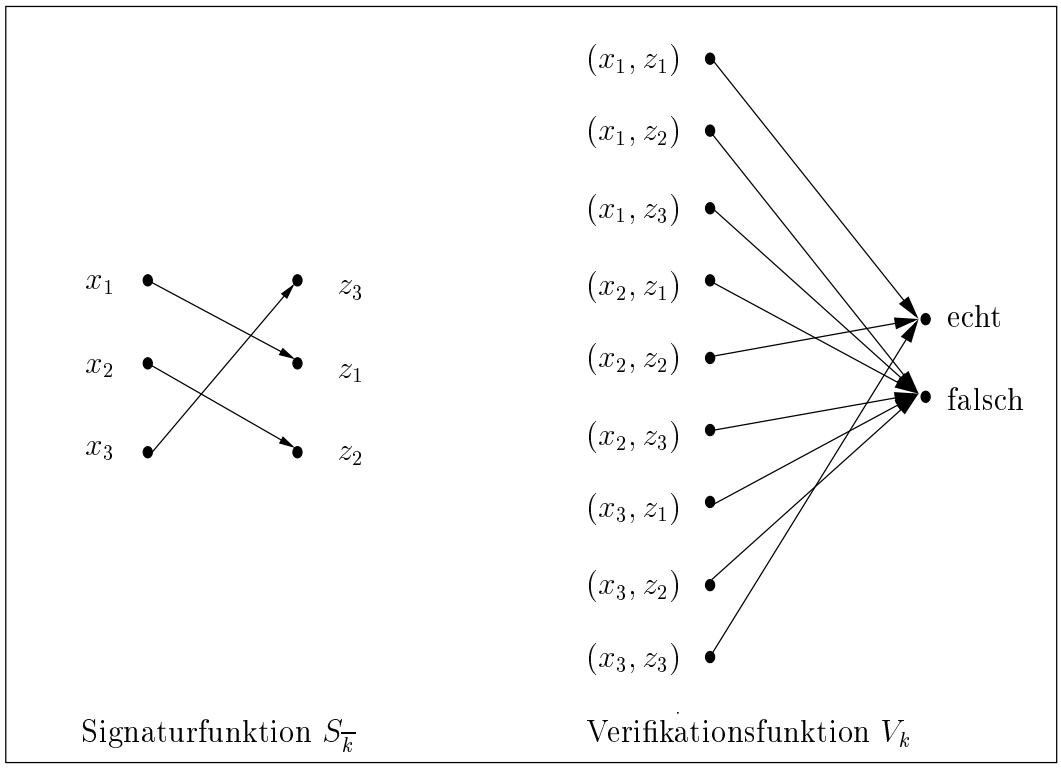


Abbildung 3.1: Signatur- und Verifikationsfunktion eines Digitalen Signaturverfahrens

Die *linke Abbildung* von Abb. 3.1 demonstriert eine *Signaturfunktion* $S_{\overline{k}}$ zu der Menge X. Jedes der Dokumente x_1, x_2 oder x_3 aus X wird durch $S_{\overline{k}}$ eindeutig auf eine Signatur z_1, z_2 oder z_3 abgebildet.
Die *rechte Abbildung* von Abb. 3.1 zeigt die korrespondierende *Verifikationsfunktion* V_k, welche mit Hilfe des öffentlichen Schlüssels k ausgibt, ob eine gegebene Signatur zu dem angegebenen Dokument von einer „Einheit", z.B. einer Person, generiert wurde oder nicht. Wird zu einem Dokument, die falsche Signatur übermittelt, so wird diese durch die Verifikationsfunktion nicht akzeptiert, da dies bedeuten könnte, daß das Dokument während der Übertragung verfälscht worden ist.
Die Funktionen (Algorithmen) $S_{\overline{k}}$ und V_k beschreiben in Abb. 3.1, wie definiert, ein **Digitales Signaturverfahren**. Die Individualität der jeweiligen Signatur wird von dem verwendeten Signaturschlüssel $\overline{k} \in \overline{K}$ und nicht nur von der verwendeten Signaturfunktion bestimmt. Man spricht deshalb von sog. *kryptographischen* Signatursystemen. Da, wie der Name schon sagt, diese Verfahren sehr stark von kryptographischen Verfahren abhängen, wird unter 3.4.1 beschrieben, wie man ein Digitales Signaturverfahren aus einem kryptographischen Verfahren herleiten kann.

Somit gelangt man zu folgender **Definition**:
Ein **Digitales Signaturverfahren** (-schema) ist ein Tupel$(X, Z, S, V, K, \overline{K})$, welches folgende Eigenschaften erfüllt:

- X ist eine endliche Menge möglicher Dokumente
- Z ist eine endliche Menge möglicher Signaturen
- Eine Signaturfunktion ist eine Abbildung $S : X \to Z$, die, angewandt auf ein Dokument $x \in X$ eine eindeutige Signatur $z = S(x)$ erzeugt.
- Eine Verifikationsfunktion ist eine Abbildung $V : X \times Z \to \{echt, falsch\}$, die für jedes Dokument $x \in X$ und für jedes $z \in Z$ verifiziert, ob die Signatur z zu dem Dokument x gültig ist.
- K ist der Schlüsselraum aller Schlüssel, die zur Signaturbildung verwendet werden.
- $\overline{K}$ ist der Schlüsselraum aller Schlüssel, die zur Verifikation von Signaturen verwendet werden.

Hierbei unterscheidet man prinzipiell zwei Digitale Signaturverfahren:
„Digitale Signaturen als Anhang" und *„Digitale Signaturen mit Dokumentenwiederherstellung"* ([AM97], S.427).

3.3 Klassifikation Digitaler Signaturen

3.3.1 Digitale Signaturen als Anhang

Digitale Signaturen als Anhang zeichnen sich dadurch aus, daß zu einem Dokument mittels eines Digitalen Signaturverfahrens die entsprechende Digitale Signatur erstellt wird und anschließend Dokument *und* Signatur an den Verifizierer übermittelt werden.
Dies ist notwendig, da bei diesen Digitalen Signaturverfahren der jeweilige Verifikationsalgorithmus das unsignierte Dokument als Eingabe benötigt.
Für einen Anwender derartiger Signaturverfahren gelten folgende Voraussetzungen:
Man bedarf eines geheimen Signaturschlüssels und muß dem Verifizierer den zugehörigen öffentlichen Verifikationsschlüssel zukommen lassen.
Der (geheime) Signierschlüssel von Bob sei $\overline{k}_B$.
Bobs (öffentlicher) Verifikationsschlüssel sei k_B.

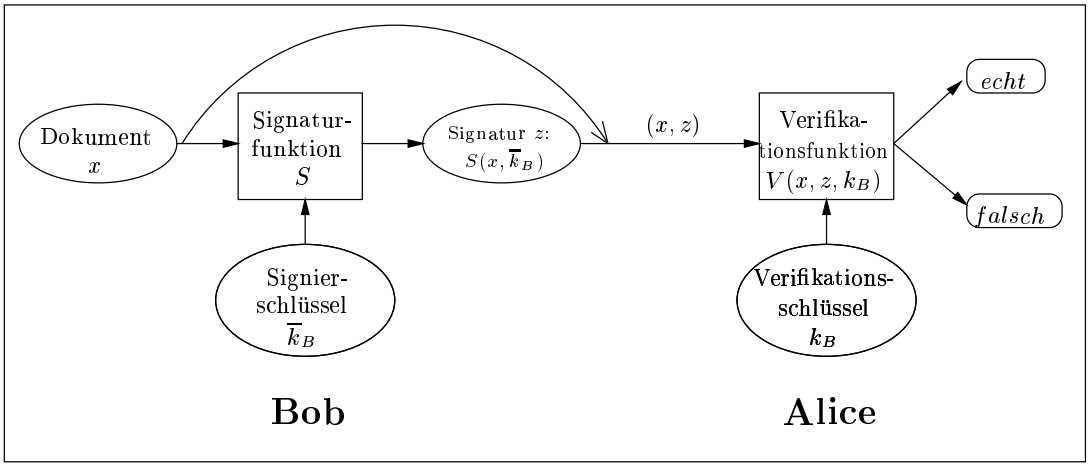


Abbildung 3.2: Digitale Signatur als Anhang

x	Dokument, $x \in X$; X: Dokumentenraum (endl. Menge v. Dokumenten)
z	Signatur, $z \in Z$; Z: Signaturraum (endliche Menge von Signaturen)
S	Signaturfunktion
V	Verifikationsfunktion
$k, \overline{k}$	Verifizier- und Signierschlüssel, $k \in K, \overline{k} \in \overline{K}$
$K, \overline{K}$	Schlüsselräume, (Signaturverifizierung und -bildung)

Bob:	signiert das Dokument x und sendet (x, z) an Alice.
Alice:	verifiziert die Signatur aus (x, z), das von Bob an Alice übermittelt wurde.
$\overline{k}_B$:	ist der (geheime) Signaturschlüssel von Bob.
k_B:	ist der (öffentliche) Verifikationsschlüssel von Bob.

Damit läuft ein Verfahren mit Digitaler Signatur als Anhang folgendermaßen ab (s. dazu Abbildung 3.2):

- **Ablauf des Signierens**
  - Bob berechnet die Signatur z zu einem Dokument $x \in X$ mit Hilfe der Signaturfunktion: $z = S(x, \overline{k}_B)$.
  - Anschließend übermittelt er Dokument und Signatur: (x, z) an Alice.
- **Ablauf der Verifikation**
  - Um sicherzustellen, daß die Signatur z zu dem Dokument x von Bob generiert wurde, muß Alice zur Verifikation:
    $u = V(x, z, k_B)$ berechnen.
  - Die Signatur ist authentisch, wenn $u = echt$ liefert.
    Es gilt:
    $$V(x, z, k_B) = \begin{cases} echt & , falls\ S(x, \overline{k}_B) = z \\ falsch, & sonst. \end{cases}$$

Beispiele für Digitale Signaturen als Anhang sind der DSA, das ElGamal- und das Schnorr-Signaturverfahren (s. [AM97] und *Kapitel 7*).

3.3.2 Digitale Signaturen mit Dokumentenwiederherstellung

Digitale Signaturverfahren mit Dokumentenwiederherstellung zeichnen sich dadurch aus, daß sie auf die Eingabe des ursprünglichen Dokuments für den Verifikationsalgorithmus verzichten können. Sie besitzen nämlich die Eigenschaft, aus der Signatur selbst das signierte Dokument mit Hilfe der Verifikationsfunktion ermitteln zu können.
Hierzu benötigt Bob wiederum seinen (geheimen) Signaturschlüssel $\overline{k}_B$ und Alice Bobs (öffentlichen) Verifikationsschlüssel k_B.
Abb. 3.3 veranschaulicht ein Digitales Signaturverfahren mit Dokumentenwiederherstellung. Angenommen Bob möchte ein Dokument x, signiert an Alice übermitteln, so daß Alice anhand der Signatur feststellen kann, ob das Dokument von Bob signiert wurde und, daß es während der Übermittlung nicht verändert wurde, so müssen Bob und Alice sich an folgenden Ablauf halten:

- **Ablauf des Signierens**
  - Das Signieren des Dokuments x läuft prinzipiell genauso wie bei „Digitalen Signaturen als Anhang" ab. Bob berechnet: $z = S(x, \overline{k}_B)$.
  - Der Unterschied liegt einzig und allein daran, daß Bob anschließend nur die Signatur z an Alice übermitteln muß.

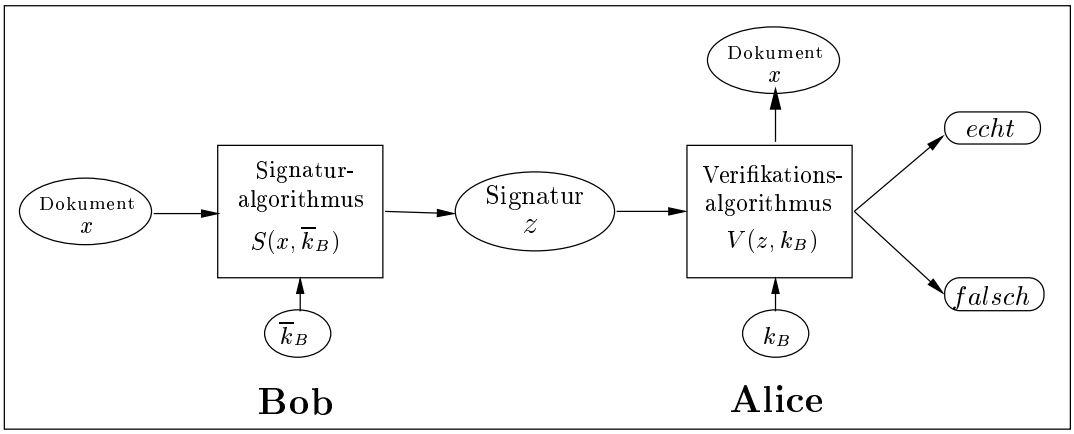


Abbildung 3.3: Digitales Signaturverfahren mit Dokumentenwiederherstellung

- **Ablauf der Verifikation**

  - Um sicherzustellen, daß die Signatur z von Bob generiert wurde, berechnet Alice: $x = V(z, k_B)$, wobei x ein Dokument darstellt.

  - Gilt $x \in X$, so liegt das ursprünglich signierte Dokument vor und die Signatur wird als gültig akzeptiert.
    Ist $x \notin X$, so weist Alice die Signatur als nicht authentisch zurück.

Beispiele für Digitale Signaturverfahren mit Dokumentenwiederherstellung sind das RSA-, Rabin-, Nypberg-Rueppel-Public-Key Signatur-Schema [AM97].

Abschließend kann man noch bemerken, daß Digitale Signaturen mit Dokumentenwiederherstellung meist zum Signieren von Dokumenten einer festen Länge angewandt werden, während Digitale Signaturen als Anhang, meist zum Signieren Dokumente beliebiger Länge verwendet werden. Möchte man Digitale Signaturen mit Dokumentenwiederherstellung anwenden, so gibt es die Möglichkeit, „Digitale Signaturen als Anhang" dementsprechend umzuwandeln. Dazu wendet man auf ein Dokument beliebiger Länge wie unter *Kapitel 3* beschrieben, zuvor eine Einweg-Hashfunktion an, um dieses auf eine feste (und kürzere) Länge zu bringen. Auf diesen Wert fester Länge kann man daraufhin das entsprechende Signaturverfahren mit Dokumentenwiederherstellung anwenden. Die andere Möglichkeit, ein Dokument beliebiger Länge in entsprechende Blöcke gleicher Länge zu unterteilen, um diese Blöcke dann mittels „Signaturverfahren mit Dokumentenwiederherstellung" zu signieren, wird in *Kapitel 3* aufgrund von Sicherheitsmankos abgelehnt [AM97].

3.4 Wichtige Grundvoraussetzungen für Signaturverfahren

Ebenso wie Kryptosysteme sollten Digitale Signaturverfahren Grundvoraussetzungen wie die Korrektheitseigenschaft und Berechnungseffizienz erfüllen, um effizient und sicher sein zu können (s. dazu *Kapitel 2*). Dazu sollte man bedenken, daß Digitale Signaturen, wie bereits erwähnt, von Public-Key-Verfahren hergeleitet werden können. Im folgenden wird nun zunächst dargestellt, in welchem Zusammenhang Digitale Signaturverfahren und Public-Key-Kryptosysteme stehen, um daraufhin die Sicherheitsmaßnahmen, die bei Digitalen Signaturen vorausgesetzt werden, einleuchtender darlegen zu können.
Kryptographische Signaturverfahren werden, wie es der Name schon andeutet, sehr stark von kryptographischen Verfahren beeinflußt. Um den Zugang zu kryptographischen Signaturverfahren zu erleichtern, wurden in *Kapitel 2* bereits kryptographische Verfahren vorgestellt, um nun aufzeigen zu können, wie man aus Public-Key-Verfahren kryptographische Signaturschemata herleiten kann.

3.4.1 Herleitung Digitaler Signaturen von Public-Key-Verfahren

Wie bereits in der Einleitung erwähnt, möchte man Digitalen Signaturen durch eine Geheiminformation eine Art „eindeutige Kennung" zuweisen, die es ermöglicht, eine Signatur einer ganz bestimmten Person bzw. Einheit zuzuordnen. Diese Geheiminformation liegt z.B. in Form eines geheimen Schlüssels vor, den nur die besagte Einheit kennt. Dabei gibt es die Möglichkeit, aus *Public-Key-Systemen* ein Signatursystem derart zu kreieren, daß mittels des Geheimschlüssels signiert und mittels des öffentlichen Schlüssels eine Signatur verifiziert werden kann. Dies basiert auf der Eigenschaft von Public-Key-Systeme, *kommutativ* zu sein: Ver- und Entschlüsselungsfunktion können vertauscht werden und liefern wiederum dasselbe Ergebnis, d.h. es gilt: $D(E(x,k),\overline{k}) = E(D(x,\overline{k}),k) = x$.
Die *Umwandlung eines Public-Key-Verfahrens zu einem kryptographischen Digitalen Signaturverfahren* kann man somit wie folgt beschreiben:
Prinzipiell wird der Ver- und Entschlüsselungvorgang bei Public-Key-Verfahren so vertauscht, daß „Bob" empfangene Dokumente nicht mit seinem privaten Schlüssel entschlüsselt, sondern er verwendet sozusagen die „Entschlüsselungsfunktion" und den privaten Dechiffrierschlüssel zum Signieren von Dokumenten. Alice hat im Gegenzug dazu die Aufgabe, die ihr übermittelten, signierten Dokumente mittels der bisherigen „Verschlüsselungsfunktion" und dem öffentlichen Schlüssel, (dem bisherigen Chiffrierschlüssel) auf Authentizität zu prüfen, d.h. die Signatur zu verifizieren. Salopp formuliert könnte man sagen es wird:

Der Dechiffrierschlüssel zum	$\rightarrow$	Signaturschlüssel
Die Dechiffrierfunktion zur	$\rightarrow$	Signaturfunktion
Der Chiffrierschlüssel zum	$\rightarrow$	Verifikationsschlüssel und
Die Chiffrierfunktion zur	$\rightarrow$	Verifikationsfunktion.

Somit erhält man prinzipiell aus einem Public-Key-Verfahren ein Digitales Signaturverfahren (mit Dokumentenwiederherstellung). Hierbei sollten nun folgende Anforderungen der Funktionen erfüllt sein.

3.4.2 Anforderungen an Digitale Signaturverfahren

Wie auch bei Kryptosystemen sollten Digitale Signaturverfahren folgende Eigenschaften aufweisen, um bei deren Anwendung Korrektheit und Sicherheit gewährleisten zu können:

- Die Signaturfunktion S und die Verifikationsfunktion V sollten einfach zu berechnen sein, um das Erstellen der Signatur in einem annehmbaren, zeitlichen Rahmen halten zu können. (*Berechnungseffizienz*)
- Die Schlüssel $k \in K$ und $\overline{k} \in \overline{K}$ sollten derart gewählt werden, daß gilt: $\forall x \in X$: $V(S(x, \overline{k}), k) = x$. (*Korrektheitseigenschaft*)

Zudem sollte eine noch „stärkere" Anforderung erfüllt sein:

- Ohne den Signaturschlüssel $\overline{k}$ zu kennen, sollte es unmöglich sein, $S(x, \overline{k})$ zu berechnen.
- Dies bedeutet, auch wenn der Verifikationsschlüssel k, der Verifikationsalgorithmus und sogar mehrere mit dem Signaturschlüssel $\overline{k}$ signierte Dokumente bekannt sind, muß es für einen Widersacher mit immensem Aufwand verbunden sein, ein signiertes Dokument zu finden, das von V unter k als echt anerkannt wird. (*Fälschungssicherheit*)

Zusammenfassend kann man feststellen, daß man durch das Umkehren eines sicheren Kryptosystems im Prinzip ein Digitales Signaturschema vorfindet. Dabei muß jedoch beachtet werden, daß die Verschlüsselungs- bzw. Signaturfunktionen die Trapdoor-Eigenschaften aufweisen, um ein Mindestmaß an Sicherheit vor Fälschungen bieten zu können. Bevor jedoch in *Kapitel 5* geschildert wird, wie man *Fälschungssicherheit* bei Digitalen Signaturverfahren erzielen kann, wird im folgenden Kapitel erläutert, wie man durch den Einsatz von Hashfunktionen die *Berechnungseffizienz* Digitaler Signaturschemata erhöhen kann.

Kapitel 4

Effizienz Digitaler Signaturen

In der Praxis hat es sich herausgestellt, daß Public-Key-Verfahren und daraus abgeleitete digitale Signaturverfahren häufig nicht sehr effizient sind. Vor allem bei deren Anwendung auf lange Texte bzw. Dokumente benötigen diese Verfahren immens viel Zeit. Die Leistungsfähigkeit, die auch für lange Texte sinnvoll wäre, kann durch den bisher beschriebenen Signaturvorgang leider nicht in dem gewünschten Maße erreicht werden. Eine Möglichkeit jedoch, die Effizienz kryptographischer Signaturverfahren zu erhöhen, eröffnet sich durch die Verwendung sog. *kryptographischer Hashfunktionen*, s. Definition 4.2. Welche Eigenschaften zeichnen derartige Funktionen aus?

4.1 Allgemeiner Hintergrund zu Hashfunktionen

Hashing, auch bekannt als Streuspeicherverfahren, ist ursprünglich eine Methode zum dynamischen Verwalten von Daten, wobei die Daten mittels eines Schlüssels angesprochen werden. Dies ermöglicht es, Informationen schnell abzuspeichern und wiederzufinden. Die gleiche Aufgabenstellung (Einfügen, Ändern, Löschen von Daten) könnte auch durch die Verwendung von Suchbäumen erreicht werden, wobei jedoch die Komplexität beim Suchen und Einfügen (im worst- und average-case) O(log n) beträgt, während sich der mittlere Aufwand beim Hashing unter bestimmten Voraussetzungen auf O(1) beläuft, d.h. konstant ist. [Sch92] Dies wird dadurch erreicht, daß die Schlüssel, die auf die entsprechenden Datensätze verweisen, nicht sequentiell in eine Tabelle eingetragen werden, sondern man verwendet eine sog. *Hashfunktion*, die angewandt auf einen solchen Schlüssel als Ergebnis einen Index, d.h. eine Adresse in der Hashtabelle liefert. Diese Hashtabelle ist ein Array von Zeigern, welche auf die entsprechenden Datensätze verweisen.
Hashfunktionen können nicht nur bei der Speicherverwaltung von Daten genutzt werden, sondern sie werden auch in kryptographischem Umfeld als sog. *kryptographische Hashfunktionen* angewandt. Derartige Hashfunktionen kann man

zunächst daran unterscheiden, ob sie lediglich vom Eingabetext oder zudem von der Verwendung eines Schlüssels abhängen. Erstere Verfahren dienen meist der Wahrung der Datenintegrität, d.h. der Erkennung, ob unbefugte Manipulationen an dem übermittelten Text stattgefunden haben oder nicht und werden folglich häufig als *Message Detection Codes* (MDC) bezeichnet.
Schlüsselabhängige Hashverfahren, welche symmetrische Schlüssel verwenden, dienen meist als sog. *Message Authentication Codes* (MAC) dazu, die Authentizität von Nachrichten sicherzustellen, da man anhand des verwendeten Schlüssels zurückverfolgen kann, von wem der Hashwert stammt (s.u.).
Schlüsselabhängige Hashverfahren mit asymmetrischen Schlüsseln zählen zu den kryptographischen Digitalen Signaturverfahren. Anstelle einer Signatur- und Verifikationsfunktion verwenden sie eine dementsprechende Hashfunktion.
Anschließend wird nun geschildert, durch welche Eigenschaften, sich kryptographische Hashfunktionen auszeichnen und wie sie es ermöglichen, die Authentizität Digitaler Signaturen zu gewährleisten.

4.2 Kryptographische Hashfunktionen und ihre Eigenschaften

Definition: *Hashfunktion*
Eine Hashfunktion ist eine effizient zu berechnende Funktion, welche Binärstrings einer beliebigen Länge auf Binärstrings einer festen Länge, auf sog. *Hashwerte* abbildet.
Anm.: Die Eigenschaft, Eingaben beliebiger Länge auf Werte fester Länge abzubilden, nennt man *Kompressionseigenschaft* der Hashfunktionen.

Allgemein zielt eine *kryptographische Hashfunktion* darauf ab, zu einem vorgegebenen Text x eine kompakte, aber dennoch repräsentative Darstellung $H(x)$ zu liefern. Dabei bietet die vorgegebene Länge der Hashwerte (in der Praxis meist 160 Bit) einige Vorteile:
Viele Signaturverfahren, wie z.B. das RSA-Signaturschema (s. *Kapitel 7*), können nur Eingaben fester Länge verarbeiten. Ohne Verwendung einer Hashfunktion müßten somit Dokumente, die größer sind, in mehrere Blöcke aufgeteilt werden, die daraufhin jeweils einzeln signiert und übermittelt werden müßten. Angreifer könnten hierbei evtl. durch Vertauschen der signierten Blöcke eine „nicht-selektive Fälschung"[1] (s. *Kapitel 5*) vornehmen und zumindest, für einige Verwirrung sorgen. Dies kann durch den Einsatz von Hashfunktionen unterbunden werden, da

[1] Dies bedeutet: Der Gegner kann zumindest für irgendein Dokument, welches vom rechtmäßigen Unterzeichner nicht schon selbst signiert wurde, die zugehörige Digitale Signatur bestimmen, wobei er so gut wie keinen Einfluß auf die Wahl des Dokuments hat, für das die Fälschung gelingt. In diesem Falle könnte z.B. ein Dokument durch Vertauschen der Blöcke einen ganz neuen Sinn erhalten.

eine Hashfunktion auf das gesamte Dokument angewandt wird und somit eine Unterteilung in Blöcke verhindern kann.
Ein weiterer Vorteil von Hashfunktionen, ist die Datenintegrität (s. Def. *Kapitel 1*). Wird z.B. ein Text von Ort A nach Ort B übertragen, so hat man in der Regel an Ort B keinerlei Möglichkeit festzustellen, ob der Text während der Übertragung verändert wurde oder nicht. Übermittelt man zusätzlich zu dem eigentlichen Text, den zugehörigen Hashwert, so kann der Empfänger selbst den Hashwert zu dem Text berechnen und anschließend mit dem übermittelten Hashwert vergleichen. Sind beide Werte identisch, so wurde die Datenintegrität gewahrt, weisen sie jedoch eine Diskrepanz auf, so muß von einer Verfälschung des ursprünglichen Textes ausgegangen werden. Eine wichtige Voraussetzung ist natürlich, daß der Hashwert über einen gesicherten Kanal[2] übertragen wird, d.h. z.B. verschlüsselt übermittelt wird, da ansonsten Angreifer auch diesen Wert entsprechend dem Dokument verfälschen könnten und der Empfänger somit nichts von der Fälschung ahnen würde.
Anmerkung: Eine grundlegende Anforderung an Hashfunktionen ist die sog. *Berechnungseffizienz*. Diese besagt, daß es zu gegebenem x leicht sein sollte, $H(x)$ zu berechnen. Dies ist einleuchtend, da man die Effizienz, die man durch die Verwendung von Hashfunktionen gewinnt, nicht durch einen dementsprechenden Berechnungsaufwand wieder einbüßen möchte.

Bisher konnte festgestellt werden, daß kryptographische Hashfunktionen zum einen dazu verwendet werden, die Effizienz von Verfahren zu steigern, zum anderen zielen sie darauf ab, die Integrität digitalisierter Daten sicherzustellen. Im folgenden werden zunächst Hashfunktionen mit symmetrischen Schlüsseln (MACs) vorgestellt, da sie, bei gegenseitigem Vertrauen der Kommunikationspartner, anstelle der aufwendigeren Digitalen Signaturverfahren eingesetzt werden können, um Authentizität und Integrität von Dokumenten zu gewährleisten.
Im Anschluß daran wird die Verwendung schlüsselloser, kryptographischer Hashfunktionen (MDCs) (s. 4.2) bei Digitalen Signaturverfahren vorgestellt. Da sie es ermöglichen, das ursprünglich zu signierende Dokument, auf wesentlich kleinere Werte abzubilden, werden sie häufig als *Message Digest* (engl.: Nachrichtenextrakt) oder *digitaler Fingerabdruck* bezeichnet. In *Abschnitt 4.2.2* wird geschildert wie deren positive Eigenschaften (Erhalt von Datenintegrität und Effizienzsteigerung) in die Signaturverfahren eingebaut werden.

[2]Man spricht von einem *gesicherten Kanal*, wenn Nachrichten oder Daten, die über diesen Kanal übertragen wird, weder von Unbefugten eingesehen, noch manipuliert werden können.

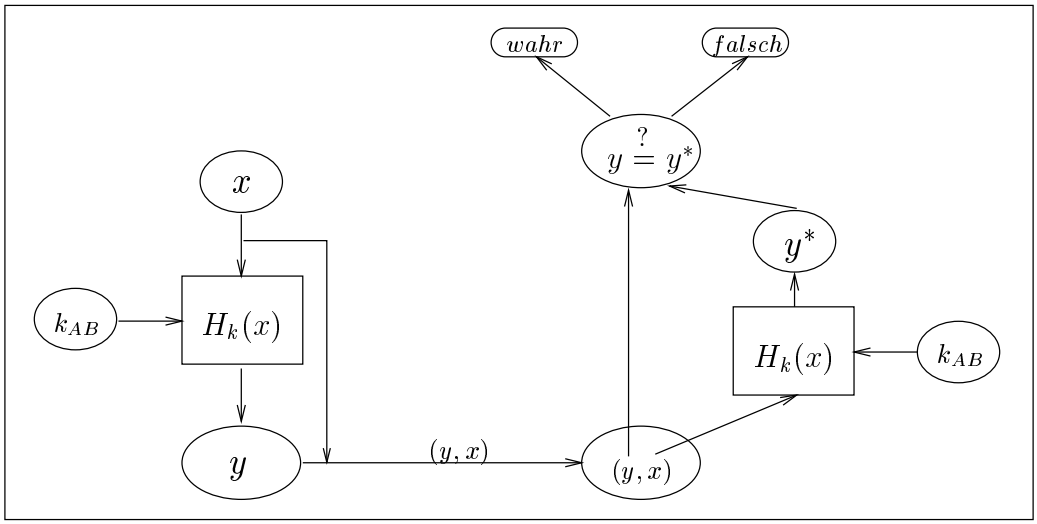


Abbildung 4.1: Entstehung eines MACs (Message Authentication Codes)

4.2.1 Message Authentication Codes: MACs

Wie bereits auch unter *Kapitel 3* erwähnt, handelt es sich bei MACs um Hashfunktionen, die bei ihren Berechnungen symmetrische Schlüssel verwenden und ebenso wie Digitale Signaturen dazu dienen, Authentizität von Dokumenten zu gewährleisten. Im Gegensatz zu Digitalen Signaturen eignen sich dieses Message Authentication Codes jedoch nicht zur Gewährleistung von Verbindlichkeit gegenüber Personen, die nicht am eigentlichen Dokumentenaustausch teilnehmen, da beide MAC-Kommunikationspartner denselben Schlüssel verwenden und somit für einen Außenstehenden nicht ersichtlich ist, wer von den beiden eine Unterschrift geleistet hat. Unter Abb. 4.1 wird der Ablauf bei der Bildung von MACs deutlich:
Bob möchte das Dokument x vor unbefugten Veränderungen während der Übermittlung zu Alice schützen. Dazu berechnet er mittels der Hashfunktion H und dem symmetrischen Schlüssel k_{AB} von Alice und Bob $y = H(x, k_{AB})$ den MAC-Hashwert y.
Anschließend übermittelt er (über einen gesicherten Kanal) Dokument und MAC-Hashwert: (x, y) an Alice.
Alice überprüft, ob die Datenintegrität des erhaltenen Dokuments x, indem sie selbst zu x den MAC-Hashwert $y^* = H(x, k_{AB})$ bestimmt und mit dem übermittelten y vergleicht. Gilt $y^* = y$, so wurden die Daten nicht verändert. Zudem kann die Nachricht nur von Bob stammen, da außer Alice nur Bob den Schlüssel k_{AB} kennt, immer vorausgesetzt natürlich, daß dieser Schlüssel wie bei symmetrischen Kryptosystemen über authentisiertem Wege ausgetauscht wurde.

Zur Sicherheit von MACs:
Damit man nicht für jeden einzelnen Kommunikationsaustausch zwischen Alice und Bob einen neuen Schlüssel benötigt, sollte die Hashfunktion H für jeden Schlüssel k folgende *Berechnungsresistenz* aufweisen:
Angenommen es sei nicht k, dafür jedoch eine Reihe von Dokument-,MAC-Paaren $(x_1, H(x_1, k)), ..., (x_n, H(x_n, k))$ die mit Hilfe von k generiert wurden, bekannt. Dann ist es mit einem immensem Aufwand verbunden, ein Paar $(x, H(x, k))$ zu bestimmen, so daß x mit keinem der Dokumente $x_1, ..., x_n$ übereinstimmt. (Dabei ist es bedeutungslos, ob der ermittelte Hashwert $H(x, k)$ bereits unter den genannten Hashwerten vorkommt oder nicht.)

Ist nun eine Hashfunktion *berechnungsresistent*, so ist es einem Gegner nicht möglich, ein Dokument bzw. eine Nachricht x an Alice zu übermitteln, die sie als authentisch anerkennt. Die Geheimhaltung des Schlüssels k ist hierbei, wie auch bei Kryptosystemen eine grundlegende Voraussetzung.
Es ist einleuchtend, daß sich ein Angriff gegen eine Hashfunktion mit Verwendung eines Schlüssels schwieriger darstellt, als gegen eine Hashfunktion ohne Schlüssel. Sucht man bei letzteren nach einer Kollision (s.u.), so kann man meist durch direkten Wertevergleich erkennen, ob man eine Kollision gefunden hat. Bei Text-MAC-Paaren (x, y) muß der Gegner meist einen Fälschungsversuch mit diesem Paar unternehmen, bis er aufgrund der Reaktionen der betroffenen Parteien erfährt, ob y mit dem gesuchten Hashwert $H(x, k)$ übereinstimmt.

Da die Anwendung dieser Message Authentication Codes, wie bereits erwähnt, keine Verbindlichkeit gegenüber Dritten gewährleisten kann, d.h. zwei Kommunikationspartner können nur einander beweisen, daß der jeweils andere der Urheber einer Signatur ist, jedoch nicht gegenüber einem „unbeteiligten" Dritten. Setzt man dies jedoch voraus, so muß man zur Wahrung von Authentizität auf Digitale Signaturverfahren zurückgreifen, die u.U. aufwendiger sind, dafür aber gewünschte Verbindlichkeit gegenüber Dritten aufweisen können.

4.2.2 Erstellen Digitaler Signaturen unter Verwendung einer Hashfunktion

Im Prinzip geht es bei der Erstellung einer Digitalen Signatur mit Hilfe einer Hashfunktion darum, nicht das Dokument direkt zu unterzeichen, sondern den dazugehörigen Hashwert. Dieses Vorgehen führt durch die Kompressionseigenschaft (s. 4.2) zum einen zu einem geringeren Berechnungsaufwand der Signatur und zum anderen zu einer Reduktion der Bandbreite, die zur Übermittlung der Signatur benötigt wird. Somit erhält man insgesamt kürzere Berechnungs- und Übertragungszeiten und gelangt folglich zu einer erhöhten Effizienz des Signaturverfahrens.

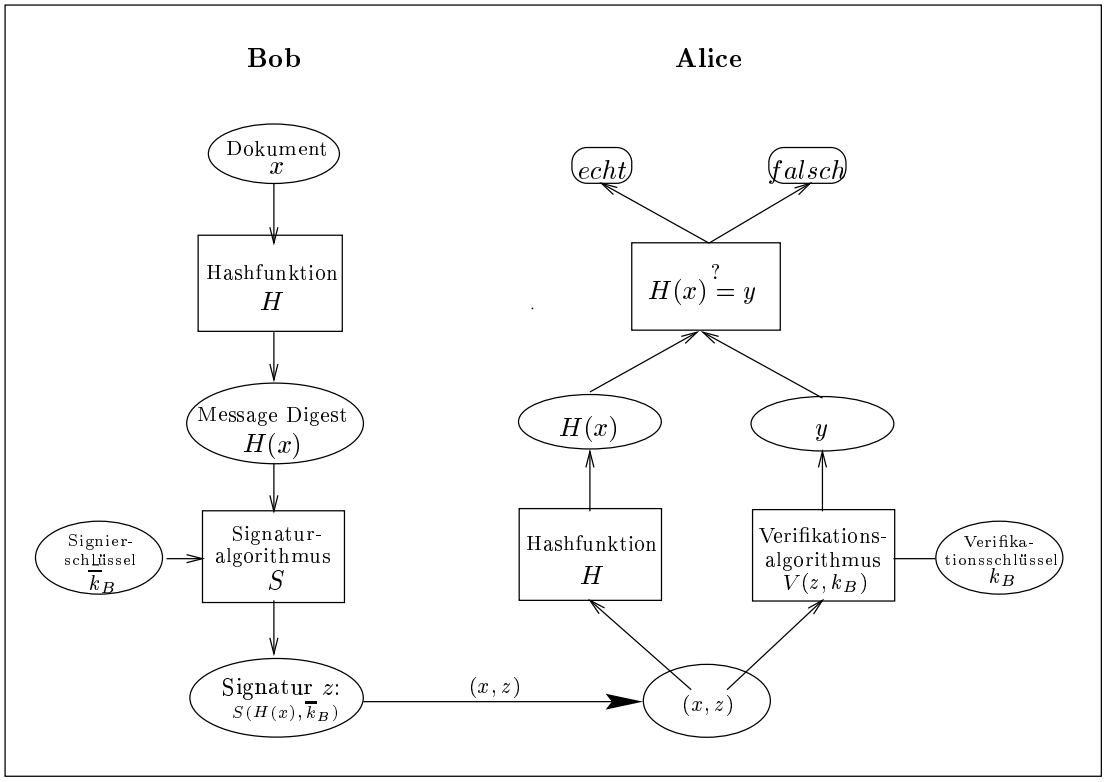


Abbildung 4.2: Digitale Signatur (als Anhang) unter Verwendung einer kryptographischen Hashfunktion

Möchte nun Bob, wie gehabt, ein Dokument signieren und an Alice übermitteln, so geht er wie folgt vor: (Siehe dazu Abb. 4.2).

1 Bob berechnet den Hashwert $H(x)$ des Dokuments x.

2 Dann signiert er den Hashwert mit seinem privaten Schlüssel, indem er folgende Berechnung vornimmt: $z = S(H(x), \overline{k}_B)$

3 Er übermittelt das Dokument und den signierten Hashwert an Alice: (x, z)

Alice überprüft die Signatur von Bob wie folgt:

1 Sie ermittelt den Hashwert von dem erhaltenen Dokument, indem sie $H(x)$ berechnet.

2 Sie verifiziert die Signatur, indem sie folgende Berechnung durchführt: $y = V(z, k_B)$

3 Sie vergleicht den von ihr selbst generierten Hashwert $H(x)$ mit dem zu verifizierenden y: $y \stackrel{?}{=} H(x)$

Stimmen diese überein, so kann Alice davon ausgehen, daß sie sich im Besitz einer gültigen Unterschrift und eines unverfälschten Dokuments befindet, sind die Werte nicht identisch, so ist Vorsicht geboten [Sti95].

Durch den Einbau einer Hashfunktion ergeben sich allerdings auch einige Gefahren, auf die man vorbereitet sein sollte.

4.3 Anforderungen an Hashfunktionen zur Signaturbildung

Damit die Sicherheit Digitaler Signaturverfahren durch die Verwendung von Hashfunktionen nicht beeinträchtigt wird, sollte man sich der Gefahren, die dadurch entstehen, bewußt werden und diese gezielt angehen.
Bei Digitalen Signaturverfahren mit Verwendung einer Hashfunktion, wird, wie unter *Abschnitt 4.2.2* beschrieben wurde, der Message Digest und nicht das Dokument direkt unterzeichnet. Ein Widersacher kann daher versuchen, zu einem zufällig gewählten Hashwert y, die dazugehörige Signatur z zu erhalten, worauf er dann ein Dokument x sucht, mit $y = H(x)$. Gelingt ihm dies, so erhält er ein gültiges Paar aus Dokument und Signatur (x, z), dem die Fälschung nicht mehr nachgewiesen werden kann.
Aufgrund dieser Fälschungsmöglichkeit ist eine grundlegende Anforderung an Digitale Signaturverfahren mit Hashfunktion die Verwendung sog. *Einweghashfunktionen*. Dies ist eine Hashfunktion, welche die sog. *Einwegeigenschaft*[3] (s. *Kapitel 2*) aufweist. Verwendet ein Digitales Signaturverfahren eine derartige Einwegfunktion, so kann diese Art von Attacke durch den dazu benötigten immensen Rechenaufwand, vereitelt werden.

Bisher wurde dem Leser verschwiegen, daß die Kompressionseigenschaft der Hashfunktionen (s. 4.2) nicht nur den positiven Effekt der Effizienzsteigerung bewirkt, sondern auch einen negativen Nebeneffekt aufweist: die Möglichkeit, daß Kollisionen auftreten können.
Definition: *Kollision*
Eine **Kollision** liegt dann vor, wenn eine Hashfunktion angewandt auf zwei unterschiedliche Eingaben x und y, denselben Hashwert $H(x) = H(y)$ liefert.

Da Hashfunktionen aufgrund der Kompressionseigenschaft, s. 4.2, Eingaben (Texte) beliebiger Länge auf Werte einer ganz bestimmten Länge abbilden und verkürzen, ist es einleuchtend, daß es früher oder später zu einer Kollision kommen muß.

[3] Diese besagt, daß es rechnerisch mit immensem Aufwand verbunden ist, zu einem gegebenem Hashwert y, ein Dokument x zu finden, so daß $H(x) = y$ gilt, wobei y die feste Länge l besitzt, $l \in N$.

Kollisionen sind fast unvermeidlich, aber dennoch in großem Maße unerwünscht, da sie sogenannte Geburtstagsangriffe (s.u.) bedingen, die es zum Ziel haben, derartige Kollisionen zu finden. Die Gefahr, die dadurch entsteht, beschreibt folgendes Szenario:
Der Angreifer ist auf der Suche nach zwei Texten, die auf denselben Hashwert abgebildet werden. Besitzt er zu einem der beiden eine gültige Signatur, so ist er automatisch auch für den anderen (nicht-legalen) Text im Besitz einer gültigen Unterschrift. Erhält Alice eine derartig „gefälschte" Signatur, so hat sie keinerlei Möglichkeit, dies festzustellen, da die Signatur laut Verifikationsalgorithmus „gültig" ist.
Daher werden im allgemeinen Hashfunktionen mit dem Ziel entwickelt, die Wahrscheinlichkeit für auftretende Kollisionen zu minimieren bzw. sogar eine gewisse *Resistenz gegen Kollsionen* zu erzielen.
Resistenz besagt in diesem Zusammenhang, daß bei praktischer Anwendung einer derartigen Hashfunktion so gut wie nie Kollisionen auftreten, auch dann nicht, wenn es ein Widersacher darauf anlegt, diese zu finden (s.u.).
Kollisionsresistenz bedeutet *nicht*, daß derartige Kollisionen überhaupt nicht existieren, sondern besagt „nur", daß es (für einen Gegner) mit einem derartig großen Berechnungsaufwand verbunden ist, derartige Kollisionen zu finden, daß es in „absehbarer" Zeit praktisch unmöglich ist.
Formal unterscheidet man noch zwischen sog. schwacher und starker Kollisonsresistenz:

Definition: *Schwache Kollisionsresistenz*
Eine Hashfunktion H ist schwach-kollisionsresistenz, wenn es zu einem gegebenen Dokument x, berechnungstechnisch[4] unmöglich ist, ein Dokument x^* zu finden, so daß gilt: $H(x) = H(x^*)$.

Definition: *Starke Kollisionsresistenz*
Eine Hashfunktion H heißt stark-kollisionsresistent, wenn es berechnungstechnisch unmöglich ist, beliebige Dokumente x und x^* zu finden, mit $x \neq x^*$, so daß gilt: $H(x) = H(x^*)$.

Anmerkung: Es ist einleuchtend, daß starke Kollisionsresistenz gleichzeitig schwache Kollisionsresistenz bedingt.

[4] berechnungstechnisch meint, mit den zur Verfügung stehenden technischen Mitteln, wie z.B. Berechnungsressourcen, Speicherkapazitäten etc.

Wie kann ein Widersacher nun bei den sog. **Geburtstagsangriffen** vorgehen?
Sei nun eine Hashfunktion $H : X \to Y$, wobei X und Y endliche Mengen, mit $\mid X \mid \geq 2 \mid Y \mid$. Sei $\mid X \mid = m$ und $\mid Y \mid = n$. Es ist leicht zu sehen, daß es mindestens n Kollisionen geben muß, die Frage ist, wie der Widersacher diese finden kann.
Eine relativ einfache Möglichkeit besteht durch besagte *Geburtstagsangriffe*. Diese lehnen sich an das sog. *Geburtstagsparadoxon* an, welches besagt, daß in einer Gruppe von 23 Personen mit einer Wahrscheinlichkeit von 50% mindestens zwei davon am gleichen Tag Geburtstag haben. Die Frage ist nun, wieviele zufällige Texte ein Angreifer auf Kollisionen untersuchen muß, um mit 50%iger Wahrscheinlichkeit auf eine Kollision zu stoßen? (Der genaue Bezug hierzu wird im folgenden klarer werden.)
Derartige Geburtstagsangriffe bedienen sich der „Brute-Force-Methode", die durch systematisches Durchprobieren (aller bzw. einiger zufälliger) Texte, versucht, derartige Kollsionen zu ermitteln. Hierbei gibt es prinzipiell zwei Möglichkeiten:

1 Ein vorgegebenes Ziel des Widersachers kann sein, zu einem gegebenem Hashwert $H(x)$, einen Text x^* zu finden, so daß gilt: $H(x^*) = H(x)$. Mit der genannten Brute-Force-Methode wählt er k zufällige, unterschiedliche Elemente $x_1, ..., x_k \in X$ aus, berechnet dazu die zugehörigen $y_i = H(x_i)$, $1 \leq i \leq k$ und stellt fest, ob eine Kollision stattgefunden hat.

2 Eine Abwandlung des bereits genannten Brute-Force-Angriffs, ist ein weniger spezifischer und daher einfacherer *Geburtstagsangriff*: Der Angreifer sucht hierbei nicht nach einem Text zu einem *speziellen Hashwert*, sondern nach *zwei* Texten x und x^*, so daß $H(x)$ und $H(x^*)$ dieselben Werte liefern. Durch die Abschwächung der Suchbedingung erhöht sich für ihn die Wahrscheinlichkeit, eine Kollision zu finden. Wie lange muß nun ein Hashwert sein, um einen derartigen Angriff unterbinden bzw. zumindest aufgrund eines entsprechenden Zeitaufwandes unmöglich zu machen? Dazu kann man bestimmte Abschätzungen durchführen.
Sei wiederum eine Hashfunktion $H : X \to Y$, wobei X und Y endliche Mengen, mit $\mid X \mid \geq 2 \mid Y \mid$. Sei $\mid X \mid = m$ und $\mid Y \mid = n$, wobei Y die Menge aller möglichen Hashwerte darstellt und X die Menge aller Texte.
Um das Risiko einer Kollision möglichst klein halten zu können, sollte man eine Hashfunktion wählen, die bei Anwendung auf die möglichen Eingaben, möglichst zufällig, d.h. gleichverteilt, auf die Hashwerte der Menge Y streut. Deshalb geht man bei folgenden Wahrscheinlichkeitsberechnungen davon aus, daß ein beliebiges $x \in X$ auf einen Hashwert $y \in Y$ mit der Wahrscheinlichkeit $\frac{1}{n}$ abgebildet wird. Zudem sei $\mid H^{-1}(y) \mid \approx \frac{m}{n}$, $\forall y \in Y$.[5]

[5]Diese Annahme bedeutet, daß wenn die inversen Bilder der Hashwerte nicht mit annähernd den gleichen Häufigkeiten auftreten, so steigt die Wahrscheinlichkeit, daß eine Kollision auftritt. Gesucht wird jedoch eine „untere Schranke" für die Wahrscheinlichkeit eine Kollision zu finden.

Im folgenden wird zunächst die Wahrscheinlichkeit berechnet, mit der keine Kollisionen auftreten, um im Anschluß durch Bestimmung der Gegenwahrscheinlichkeit, das Auftreten von mindestens einer Kollision berechnen zu können, denn es gilt:
P[Es tritt mindestens eine Kollision auf] = 1 - P[Es treten keine Kollisionen auf][6]

Man untersucht nun folgende *Aufgabenstellung*:
Gesucht wird: die Wahrscheinlichkeit, daß zu k zufällig gewählten „Elementen" $x_1, ..., x_k \in X$, die zugehörigen Hashwerte $y_1, ..., y_k \in Y$ unterschiedlich sind, d.h. keine Kollisionen aufweisen.

Berechnung: Die y_i's seien folgendermaßen geordnet: $y_1, ..., y_k$.
Die Wahl von y_1 sei beliebig.
Die Wahrscheinlichkeit, daß $y_2 \neq y_1$ ist dann $(1 - \frac{1}{n})$.
Die Wahrscheinlichkeit, daß y_3 unterschiedlich von y_2 und y_3, wobei bereits $y_2 \neq y_1$ gilt, ist dann $(1 - \frac{2}{n})$, etc.
Folglich kann man die Wahrscheinlichkeit, daß keine Kollision auftritt folgendermaßen abschätzen:

$$\left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right) ... \left(1 - \frac{k-1}{n}\right) = \prod_{i=1}^{k-1}\left(1 - \frac{i}{n}\right)$$

Ist nun x eine kleine reelle Zahl, dann gilt[7]: $1 - x \approx e^{-x}$.
Dann beträgt P[Es tritt keine Kollision auf]:

$$\prod_{i=1}^{k-1}\left(1 - \frac{i}{n}\right) \approx \prod_{i=1}^{k-1} e^{\frac{-i}{n}} = e^{\frac{-k(k-1)}{n}}$$

Folglich kann man P[Es tritt mindestens eine Kollision auf] abschätzen mit:

$$1 - e^{\frac{-k(k-1)}{n}}$$

Dies sei die Wahrscheinlichkeit: ϵ
Damit kann k (die Anzahl der zufällig zu testenden Werte, um eine Kollision finden zu können) als Funktion von n und ϵ dargestellt werden:

$$e^{\frac{-k(k-1)}{n}} \approx 1 - \epsilon$$

$$\frac{-k(k-1)}{n} \approx \ln(1 - \epsilon)$$

[6] P:=Wahrscheinlichkeit von

[7] Diese Abschätzung ist hergeleitet von $e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!}...$, wobei die hinteren Terme bei kleinem x vernachlässigbar sind.

$$k^2 - k \approx n \ln \frac{1}{1-\epsilon}$$

Vernachlässigt man den Term -k, dann erhält man für k:

$$k \approx \sqrt{n \ln \frac{1}{1-\epsilon}}$$

Wählt man für $\epsilon = 0.5$, d.h. für die Wahrscheinlichkeit, daß mindestens eine Kollision auftritt 50%, dann gilt:

$$k \approx 1.17\sqrt{n}$$

Dies besagt zum einen, daß die Kollisionswahrscheinlichkeit bei diesem Geburtstagsangriff nur von der Anzahl zufällig gewählter Elemente k und der Hashwertgröße n abhängt, nicht aber von der Größe m der Menge X. Dies bedeutet, daß man $1.17\sqrt{n}$ Elemente der Menge X auf eine Kollision hin prüfen muß, um auch tatsächlich mit 50%iger Wahrscheinlichkeit eine Kollision finden zu können.
Anm.:
Ein anderer Wert für ϵ führt natürlich auch zu einem veränderten konstanten Faktor, ändert jedoch nichts an der Tatsache, daß k proportional zu $\sqrt{n}$ ist.

Nach Ableitung der Kollisionswahrscheinlichkeit leuchtet nun auch der Bezug zum Geburtstagsparadoxon ein: Steht X für die Menge aller menschlichen Wesen, ist Y die Menge aller 365 Tage in einem Nicht-Schaltjahr und stellt die Hashfunktion $H(x)$ den Geburtstag einer Person x dar, so erhält man für $n = 365$ und $\epsilon = 0.5$, den Wert für $k = 22.3$. Dies bedeutet, wie eingangs erwähnt, daß unter 23 Personen mit einer Wahrscheinlichkeit von 50% mindestens zwei am gleichen Tag Geburtstag feiern, d.h. im übertragenen Sinne bei mindestens zweien davon eine Kollision auftritt.

Zur praktischen Durchführung:
Ein wesentlicher Gesichtspunkt bei der Suche nach Kollisionen ist die praktische Durchführung derselben, da diese den Zeitaufwand erheblich beeinflußt. Angenommen man wählt eine „naive" Vorgehensweise, so müßte man jeden neu berechneten Hashwert (zu einem neu gewählten Dokument) mit den Hashwerten der bereits verwendeten Dokumente vergleichen, so steigt der Aufwand pro neuer Berechnung proportional mit der Anzahl der bereits betrachteten Dokumente/Texte an, d.h. der Zeitaufwand wäre insgesamt proportional zum Quadrat der Anzahl der insgesamt zu betrachtenden Dokumente.

Seien nun eine Folge von Texten $x_1, x_2, ...$ und die zugehörigen Hashwerte $H(x_1), H(x_2), ...$ auf der Menge $\{0,1\}^l$ unabhängig gleichverteilt. Angenommen man müßte durchschnittlich $2^{\frac{l}{2}}$ Dokumente untersuchen bis man auf eine Kollision trifft, so würde bei dieser naiven Vorgehensweise ein Zeitaufwand von $(2^{\frac{l}{2}})^2 = 2^l$ auftreten.
Trägt man jedoch zu jedem Dokument den dazu berechneten Hashwert in eine Hashtabelle ein, so kann der Zeitaufwand für die Bearbeitung eines Dokuments mit dem Aufwand der Hashtabelle abgeschätzt werden, da der Aufwand eine Kollision zu finden, bei der Verwendung von Hashfunktionen mit O(1) konstant ist. Da zudem für die Hashtabelle eine Größe von $2^{\frac{l}{2}}$ ausreichend ist, so kann man Zeitaufwand und Platzbedarf insgesamt mit $2^{\frac{l}{2}}$ abschätzen.
Um den Zeitaufwand für die Kollisionssuche also möglichst gering zu halten, sollte man auf die naive Vorgehensweise verzichten und z.B. letztere Methode wählen.

Zusammenfassend kann man feststellen, daß man bei Digitalen Signaturverfahren unter Verwendung von Hashfunktionen genannte Punkte beachten sollte:
Wichtig ist zum einen, daß derartige Verfahren Einweghashfunktionen verwenden: Angenommen es gelänge einem Angreifer, zu einem beliebig gewählten Hashwert, die Signatur zu ergattern, so scheitert er spätestens aufgrund der Einwegeigenschaft daran, den dazugehörigen Text zu finden.
Desweiteren sollten die Hashfunktionen zumindest schwach kollisionsresistent sein, um Angreifern, die keinerlei Kontrolle über das ursprüngliche Dokument haben, welches der Hashfunktion unterworfen wird, standhalten zu können.
Starke Kollisionsresistenz sollte dann gewährleistet sein, wenn es einem Widersacher z.B. in Form eines Geburtstagsangriffs möglich ist, Attacken auf das Signatursystem auszuüben. Derartige Kollisionsresistenz kann derzeit ab einer Hashwertgröße von 128 bit erreicht werden. Hierzu müßte ein Angreifer über 2^{64} zufällige Eingaben testen, um mit 50%iger Wahrscheinlichkeit auf eine Kollision zu stoßen, was derzeit auch mit zur Verfügung stehenden Parallelrechnern eine große Herausforderung darstellt. Für langfristige Sicherheit empfielt das NIST[8] jedoch aufgrund der heutigen rasenden Entwicklung der Computertechnolgien bereits 160 Bit-Hashwerte ([AM97], S.492).

Kann man diese Anforderung erfüllen, so ist die Sicherheit für Digitale Signaturen mit Hashfunktionen bzgl. der Hashfunktionen gegeben, wenn man davon ausgeht, daß Geburtstagsangriffe die einfachste (und einzige) Möglichkeit darstellen, derartige Verfahren anzugreifen. Die dadurch gewonne Effizienz und die Gewährleistung von Datenintegrität führen dazu, daß Digitale Signaturverfahren (fast) immer unter Verwendung kryptographischer Hashfunktionen generiert werden.

[8] National Institute of Standards and Technology

Kapitel 5

Sicherheit Digitaler Signaturen

Die Sicherheit digitaler Signaturverfahren ist von großer Bedeutung, da nur sichere Systeme im kommerziellen Bereich Verbreitung finden. Genau dies möchte man aber mit Digitalen Signaturen erreichen: ein möglichst breites Einsatzgebiet, um (hand)schriftliche, zeit- und kostenintensive Geschäftstätigkeiten digitalisieren und somit schneller abwickeln zu können. Um Gefahren für derartige Systeme richtig einschätzen zu können, muß man sich darüber im klaren sein, welche Arten von Angriffen es gibt und wie bzw. ob man diesen Angriffen Sicherheitsmaßnahmen entgegensetzen kann.

5.1 Der Begriff: Sicherheit

Digitale Signaturverfahren sind ähnlich wie Kryptosysteme gewissen Sicherheitsattacken ausgesetzt. Das Ziel eines Gegners, ist es Digitale Unterschriften zu fälschen, d.h. Signaturen zu generieren, die von einer Einheit bzw. dem entsprechenden Verifikationsalgorithmus als „echt" akzeptiert werden. Die *Goldene Regel* für die Entwicklung von Kryptosystemen besagt, „daß man Angreifer niemals unterschätzen sollte. Man sollte immer davon ausgehen, daß dem Gegner das angewandte System bekannt ist." (s. [Kö94], S.4) Übertragen auf Digitale Signaturverfahren bedeutet dies, daß folgende *Fälschungssicherheit*, wie in *Kapitel 3* bereits angedeutet wurde, gegeben sein muß:
„Es sollte einem Gegner ohne Kenntnis des Signaturschlüssels nicht möglich sein, eine Signatur zu generieren, die vom Verifikationsalgorithmus als gültig anerkannt wird."

Hierbei kann man Angriffe nach ihren „Aussichten auf Erfolg" bewerten, d.h. anhand dieser Kriterien kann die Gefährlichkeit einer Fälschung beurteilt werden:

1 *Nicht-selektives Fälschungsvermögen* (existential forgery): Ein Gegner besitzt nicht-selektives Fälschungsvermögen, wenn er für irgendein Dokument, die zugehörige Signatur bestimmen bzw. sich beschaffen kann.

2 *Selektives Fälschungsvermögen* (selective forgery):
Bei selektivem Fälschungsvermögen hat der Fälscher einen Weg gefunden, zu selbst gewählten Dokumenten die zugehörige Signatur zu bestimmen, z.B. durch die aktive Unterstützung des Unterzeichners, ohne daß dieser die Fälschungsabsichten kennt.

3 *Uneingeschränktes Fälschungsvermögen* (total break):
Besitzt ein Widersacher uneingeschränktes Fälschungsvermögen, so kennt er entweder den geheimen Signaturschlüssel $\overline{k}$ oder er ist in Besitz eines äquivalenten Signaturalgorithmus, welcher das gleiche Input-, Output-Verhalten wie der originale Algorithmus aufweist.

Eine wichtige Erkenntnis bei der Verwendung Digitaler Signaturverfahren ist (wie auch bei Public-Key-Kryptosystemen) leider die, daß Digitale Signaturschemata keine informationstheoretische Sicherheit (s. Def. *Kapitel 2*) aufweisen.

Warum sind Digitale Signaturverfahren nicht informationstheoretisch sicher?

Bei einem Digitalen Signaturverfahren ist, wie man bereits unter *Kapitel 3* feststellen konnte, zumindest die Verifikationsfunktion, sowie der Verifikationsschlüssel öffentlich bekannt. Ein Widersacher kann nun zufällig Signaturen „wählen" und mittels der Verifikationsfunktion und des Verifikationsschlüssels testen, ob die jeweilige Signatur akzeptiert wird oder nicht. Auf diese Art und Weise ist es einem Gegner prinzipiell möglich ohne Kenntnis des geheimen Signaturschlüssels, Unterschriften zu fälschen, es sei denn das System ist komplexitätstheoretisch sicher (s. *Kapitel 2*), denn dann reicht die Rechen- und Speicherkapazität des Widersachers nicht aus, um in absehbarer Zeit, dementsprechende Signaturen zu finden, d.h. das Signatursystem zu brechen.
Dies bestätigt allerdings obige Behauptung, daß ein Digitales Signaturverfahren aufgrund seiner Beschaffenheit[1] niemals informationstheoretisch, sondern maximal komplexitätstheoretisch sicher sein kann.
Welche Angriffsmöglichkeiten sind nun auf Digitale Signaturverfahren möglich?

[1] Da Public-Key-Verfahren laut *Kapitel 2 keine informationstheoretische Sicherheit aufweisen können und Digitale Signaturen von diesen hergeleitet werden können (s. Kapitel 3), liegt es nahe, daß auch sie, wie soeben geschildert keine informationstheoretische Sicherheit aufweisen können.*

5.2 Angriffsmöglichkeiten auf Digitale Signaturen

Prinzipiell unterscheidet man bei Angriffen auf Systeme, seien es nun Krypto- oder damit verwandte Systeme, wie z.B. Digitale Signaturverfahren, zwei Möglichkeiten: Passive und aktive Angriffe.

1 *Passive Angriffe:*
Bei passiven Angriffen, sog. Lauschangriffen, werden die von Alice zu Bob übertragenen Nachrichten bzw. Signaturen „abgehört". Dies bedeutet, daß das Ziel dieses Angriffs, die Verletzung der Vertraulichkeit der Nachrichten bei der Übertragung darstellt: Nicht nur der gewünschte Empfänger erhält die übermittelten Informationen, sondern eben auch der Gegner.

2 *Aktive Angriffe:*
Bei aktiven Angriffen versucht der Gegner nicht nur durch passives Abhören an Informationen zu gelangen, sondern er möchte durch aktive Maßnahmen, die übermittelte Information (z.B. die Signatur) verändern, zielt also auf Verletzung der Datenintegrität.

Um passive Angriffe zu vereiteln versucht man z.B. mittels kryptographischer Verfahren (s. *Kapitel 2*), derartige Lauschangriffe nutzlos zu machen: Der Gegner kann somit den Kryptotext abhören, aber er kann ihn nicht verstehen. Falls jedoch der Gegner weiß, um was es sich bei dem abgehörten Kryptotext handelt (z.B. Zugriffcodes, Zahlungsanweisungen an eine Bank bzw. eine dementsprechende Signatur), so könnte er allein durch das *Wiedereinspielen*[2] der abgehörten Informationen Schaden anrichten. Dies kann z.B. durch Verwendung von Zeitstempeln, die den Zeitpunkt des Versendens bzw. Signierens einer Nachricht angeben, vereitelt werden, da der Empfänger somit erkennen kann, ob er diese Nachricht bereits erhalten hat oder nicht. Die Wahrung von Datenintegrität kann man durch den Einsatz von Hashfunktionen und dabei zu beachtenden Sicherheitsanforderungen wie unter *Kapitel 4* geschildert, erreichen. Ein weiterer wichtiger Gesichtspunkt bei der Betrachtung von Angriffen auf kryptographische Digitale Signaturverfahren, sind die unterschiedlichen Ausgangspositionen, die ein Gegner dabei einnehmen kann.

[2]Hierbei stellt sich die Frage, ob die Wiedereinspielung abgehörter Nachrichten auch noch als passiver Angriff oder bereits als aktives Eingreifen gewertet werden soll. Dies sei dem Leser selbst überlassen.

Angriffe mit unterschiedlichen Ausgangspositionen:

1 *Angriff mit bekanntem Verifikationsschlüssel* (key-only attack):
Bei dieser Art von Angriffen kennt der Gegner nur den öffentlichen Verifikationsschlüssel für Signaturen und kann z.B. mittels eines Brute-Force-Angriffes (s.u.) versuchen, Fälschungen durchzuführen.

2 *Angriff mit bekannten Signaturen* (known signature attack):
Hierbei kennt der Gegner für eine Reihe von Dokumenten, die er allerdings nicht selbst bestimmen konnte, die zugehörigen Digitalen Signaturen.

3 *Angriff bei wählbaren Dokumenten* (chosen message attack):
Bei diesen Angriffen kann sich der Gegner zu selbst bestimmten Dokumenten (zumindest für einen bestimmten Zeitraum) die zugehörigen Digitalen Signaturen beschaffen.

Da die Gefahr einer Fälschung natürlich um so größer wird, je mehr Informationen ein Kryptanalytiker vorweisen kann, da er aus diesen evtl. Rückschlüsse auf den Signaturschlüssel ziehen kann, sollte man als Unterzeichner sehr genau darauf achten, was man unterzeichnet bzw. niemals den eigenen Signaturschlüssel in irgendeiner Weise preisgeben. Ist dieser Schlüssel z.B. auf einer Chipkarte gespeichert, so sollte man diese nicht unvorsichtig jedem zugänglich aufbewahren, auch dann nicht, wenn sie zusätzlich durch die Verwendung eines Passwortes geschützt ist.
Zudem sollten die Sicherheitsanforderungen eines Digitalen Signatursystems immer auch von der Anwendung des Verfahrens abhängig gemacht werden sollte. Kann man davon ausgehen, daß ein Angreifer nur in der Lage ist, einen „Angriff mit bekanntem Verifikationsschlüssel" zu starten, so dürfte es ausreichend sein, ein Digitales Signatursystem zu generieren, welches gegen „selektive Angriffe" gefeit ist. Besitzt jedoch der Widersacher die Möglichkeit „Angriffe mit wählbaren Dokumenten" durchzuführen, so sollten bei dem jeweiligen digitalen Signaturverfahren auch gegen „uneingeschränktes Fälschungsvermögen" Sicherheitsvorkehrungen getroffen werden.
Es gibt allerdings nicht nur Angriffe auf das Digitales Signaturverfahren an sich, sondern Widersacher können das System auch dadurch angreifen, indem sie Angriffe auf das verwendete Protokoll[3] durchführen. Hierbei sollte man auf folgende zwei Angriffe vorbereitet sein, da diese sehr grundsätzliche Angriffsmethoden darstellen.

[3]Ein *Protokoll* beschreibt durch einen Satz von Regeln, eine genau festgelegte Folge von Kommunikationsschritten zwischen zwei oder mehreren Komponenten (z.B. Rechnern, Teilnehmern etc.)

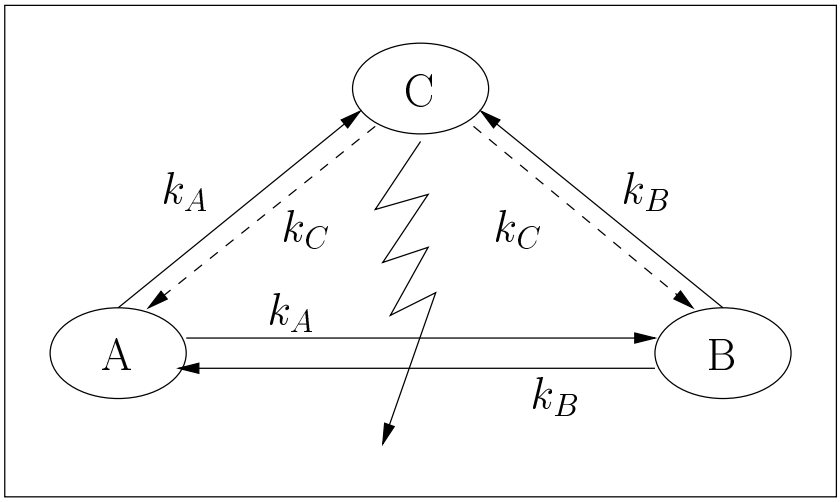


Abbildung 5.1: Man-In-The-Middle-Angriff

5.2.1 Angriffe gegen das Protokoll: Impersonations- und Brute-Force-Angriff

- *Man-In-The-Middle- oder Impersonationsangriff:*
  Beim sogenannten „Man-in-the-Middle-Angriff" (s. Abb. 5.1) schaltet sich der Widersacher „C" zwischen die Kommunikation von Alice „A" und Bob „B" ein. Wird zum Austausch der jeweils öffentlichen Verifikationsschlüssel kein authentisierter[4], sondern ein ungesicherter Kanal[5] verwendet, so kann es einem Gegner gelingen, sich gegenüber Alice als Bob und gegenüber Bob als Alice auszugeben, indem er den jeweiligen öffentlichen Verifikationsschlüssel k_A und k_B abfängt und stattdessen zu Bob und Alice seinen Verifikationsschlüssel k_C einspielt. Alice kommuniziert dann mit C und denkt es wäre Bob, Bob kommuniziert mit C und nimmt an, es wäre Alice. So kann C die Signaturen dementsprechend verändern bzw. auch die signierten Dokumente gekonnt fälschen. Alice und Bob ahnen nichts von dem Betrug. Deshalb ist es wichtig, eine *Schlüsselauthentikation* durchzuführen, d.h. zu überprüfen, ob der empfangene Verifikationsschlüssel auch von der „Einheit" stammt, die sie vorgibt zu sein. Dies kann dadurch geschehen, daß Bob und Alice über einen „authentisierten Kanal", z.B. eine Telefonleitung kommunizieren und sich die jeweiligen Verifikationsschlüssel mündlich übermitteln. Kennen sich Bob und Alice persönlich, so können sie anhand der Stimme erkennen, um wen es sich am anderen Ende der Leitung han-

[4]Ein authentisierter Kanal garantiert für die Übertragung von Daten bzw. Nachrichten Authentizität, allerdings nicht zwangsläufig Vertraulichkeit.

[5]Ein ungesicherter Kanal kann weder für Authentizität noch für Vertraulichkeit für die übertragenen Daten garantieren.

delt und können sich auf diese Art und Weise sicher sein, daß der jeweilige Schlüssel authentisch ist. Haben Alice und Bob jedoch z.B. nur schriftlichen Kontakt, so könnten sie sich z.B. durch eine *Zertifizierungsstelle* (s. *Kapitel 7*) die jeweils zertifizierten Verifikationsschlüssel des anderen beschaffen und sich somit sicher sein, daß diese Schlüssel authentisch sind und dementsprechend vor dem beschriebenen Impersonationsangriff sicher sind.

- *Brute-Force-Angriff:*
  Brute-Force-Angriffe werden sehr häufig auf Public-Key-Systeme (s. *Kapitel 2*) angewandt und stellen somit auch ein Gefahrenpotential für Digitale Signaturverfahren dar. Bei einem derartigen Angriff führt der Gegner eine vollständige „Verifikation" aller möglichen Signaturen durch, bis er eine (oder mehrere) Unterschriften findet, die den Verifikationsalgorithmus als „echt" besteht.
  Handelt es sich dabei um ein Digitales Signaturverfahren mit Dokumentenwiederherstellung, so liegt auf diese Art und Weise ein nicht-selektiver Angriff vor: Mit der gültigen Signatur ist der Angreifer (ohne Kenntnis des geheimen Signaturschlüssels) in Besitz eines gültigen Dokument-Signatur-Paares (wobei es natürlich nicht gesagt ist, daß das Dokument irgendeinen Sinn ergibt).
  Greift der Widersacher hingegen Digitale Signaturen als Anhang an, so liegt hierbei ein selektiver Angriff vor, da er zu einem gegebenen (übertragenen) Dokument x eine passende Signatur finden muß. Der Angriff stellt somit, wie bereits unter *Kapitel 2* beschrieben, eine wesentlich größere „Herausforderung" dar, da die Erfolgswahrscheinlichkeit bei einem derartigen Angriff aufgrund verwendeter Trapdoorfunktionen extrem gering ist.
  Um dem beschriebenen *nicht-selektiven Angriff* widerstehen zu können, kann man, wie folgt, sog. Redund anzfunktionen verwenden.

5.2.2 Redundanzfunktionen als „Antwort" auf nicht-selektive Fälschungen bei Digitalen Signaturen mit Dokumentenwiederherstellung

Definiton: *Redundanzfunktion*
Eine Redundanzfunktion ist eine leicht zu invertierende Funktion mit der Eigenschaft, daß in $R(x)$ reichlich Redundanz enthalten ist, selbst wenn x wenig Redundanz[6] enthält.

[6] *Redundanz* ist die Bezeichnung für die Anteile einer Nachricht, die keine Information vermitteln, also überflüssig sind.

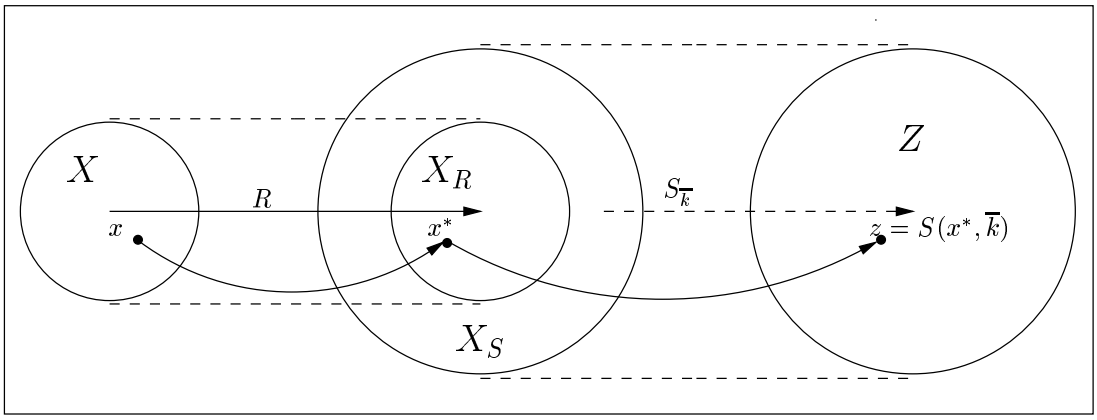


Abbildung 5.2: Überblick über ein Digitales Signaturschema mit Dokumentenwiederherstellung und Verwendung einer Redundanzfunktion

Die Signaturbildung mit Verwendung einer Redundanzfunktion verläuft dann folgendermaßen:

- Bob wendet zunächst auf das zu signierende Dokument x die Redundanzfunktion R an und berechnet $x^* = R(x)$, wobei gilt: $R : X \to X_R$.
- Anschließend signiert er x^* wie gehabt mittels der Signaturfunktion S und seinem Signaturschlüssel $\overline{k}_B$: $S(R(x), \overline{k}_B) = z$ und erhält dadurch die Signatur z. Es gilt: $S : X_R \times \overline{K} \to Z$.
- Alice verifiziert die Signatur z ebenfalls wie unter *Kapitel 3* beschrieben, indem sie die Verifikationsfunktion V auf z anwendet: $V(z, k_B) = y$.
- Liegt der berechnete Wert y im Bildbereich von R, d.h. $x^* \in X_R$, so gilt die Signatur als authentisch und Alice kann das ursprüngliche Dokument durch Anwendung der Umkehrfunktion $R^{-1}(y) = x$ wiederherstellen. Falls $x^* \notin X_R$, so weist Alice die Signatur als nicht-authentisch zurück.

Die Verwendung einer Redundanzfunktion R bewirkt, daß die Menge aller gültigen Signaturen nur noch einen Bruchteil in der Menge aller möglichen Signaturen darstellt, wodurch es einem Widersacher erschwert wird, eine gültige Signatur „zufällig" zu finden: Versucht ein Widersacher z.B. mittels des Brute-Force-Angriffs bei zufälliger Wahl einer Signatur das System anzugreifen, so ist die Wahrscheinlichkeit, hierbei Erfolg zu haben, durch die Anwendung einer Redundanzfunktion dementsprechend verringert. Liegt nämlich das hierzu durch die Verifikationsfunktion V berechnete x^* nicht im Bildbereich der Redundanzfunktion X_R, sondern z.B. „nur" in X_S, so wird die Fälschung erkannt, was ohne die

Verwendung von R nicht möglich gewesen wäre (s. Abb. 5.2).

Damit eine Redundanzfunktion auch dann, wenn die Redundanzfunktion R und deren Umkehrfunktion R^{-1} öffentlich bekannt sind, sicher sein kann, muß sie gewisse Anforderungen erfüllen.

Wichtige Anforderungen an Redundanzfunktionen:
Allgemein gilt, wie bei den bisherigen Digitalen Signaturverfahren:
Es sollte berechnungstechnisch für jemand anderen (als autorisierte Personen) mit immensem Aufwand verbunden sein, ein $z \in Z$ zu finden, so daß $V(z) \in X_R$.
Wichtig hierbei ist, daß die Redundanzfunktion in Abhängigkeit der verwendeten Signaturfunktion gewählt wird, um nicht-selektive Angriffe zu vereiteln: Angenommen dem wäre nicht so und es würde gelten: $X_R = X_S$ und R und S wären Bijektionen: $R : X \to X_R$ und $S : X_S \to Z$. Dies würde bedeuten X und Z hätten dieselbe Anzahl von Elementen. Dann würde für jedes $z \in Z$, $V(z)$ in X_R liegen und es wäre einfach, Dokumente (Texte) x und dazugehörige Signaturen z zu finden, indem man zunächst $x^* = V(z, k)$ berechnet und anschließend mittels der Berechnung $x = R^{-1}(x^*)$ das zu z zugehörige Dokument x erhält.
Stellt jedoch der Bildbereich von R nur einen Bruchteil aller möglichen zu signierenden Dokumente dar, d.h. $X_R \subseteq X_S$, so wird die Wahrscheinlichkeit wesentlich verringert, daß bei zufällig gewähltem z, selbst wenn $z \in Z$ gilt, der Verifikationsalgorithmus V die Signatur als „echt" anerkennt, da z nicht nur in X_S, sondern auch in X_R liegen muß (s. Abb. 5.2). Dies bedeutet R muß so gewählt sein, daß die Wahrscheinlichkeit zufällig ein $z \in Z$ zu finden, welches nicht nur in X_S, sondern auch in X_R liegt, so gering ist, daß es durch zufälliges Durchprobieren von Signaturen praktisch unmöglich ist.
Ein Beispiel für eine derartige Redundanzfunktion ist z.B. folgende:
Angenommen $X = \{x : \ x \in \{0,1\}^n\}$ für bestimmte positive Integer n und sei $X_S = \{t : \ t \in \{0,1\}^{2n}\}$.
Dann definiere $R : X \to X_S$, wobei $R(x) = x \parallel x$, wobei $\parallel$ eine Konkatenation darstellt, d.h. $X_R = \{x \parallel x, \ x \in X\} \subseteq X_S$. Für große Werte für n, ist dann $\mid X_R \mid / \mid X_S \mid = (\frac{1}{2})^n$, d.h. ein vernachlässigbarer Wert. Dies bedeutet, daß für einen Gegner die Wahrscheinlichkeit, eine zufällig gewählte Signatur z zu finden, so daß $V(z, k)$ in X_R liegt vernachlässigbar klein ist und somit ein dementsprechender Angriff mit großer Wahrscheinlichkeit vereitelt werden kann.

5.3 Wahl der Schlüssellänge bei Digitalen Signaturverfahren

Die Schlüssellänge stellt einen wesentlichen Aspekt bei der Beurteilung der Sicherheit von Digitalen Signaturen dar, da es für einen Widersacher unmöglich sein sollte, zu einem öffentlichen Verifikationsschlüssel den zugehörigen, geheimen Signaturschlüssel zu finden. Sind die Schlüssel bei der Erstellung von Unterschriften zu kurz, so könnten Widersacher diese z. B. mittels geschilderter *Brute-Force-Methode* in absehbarer Zeit ermitteln und damit Unterschriften fälschen.
Aufgrund der engen Verwandtschaft von Digitalen Signaturen zu Public-Key-Kryptosystemen (s. *Kapitel 3*) gilt für die Sicherheit dieser Systeme prinzipiell das Gleiche wie bei diesen Kryptosystemen, d.h. derzeit gelten Schlüssellängen mit 1024 Bit-Länge bei asymmetrischen Schlüsseln als sicher. Man sollte jedoch stets bedenken, daß man eine konkrete, sichere Schlüssellänge niemals hundertprozentig nennen kann. Die Dauer der Verwendung eines Schlüssels ist immer mitzuberücksichtigen, d.h. wie lange die Sicherheit eines Dokuments durch diesen Schlüssel mindestens gewährleistet sein sollte. Erst nach Betrachtung des zu verwendenden (Signatur- bzw. Verifikations-) Algorithmus und den Überlegungen bzgl. der geplanten Verwendung kann man die Schlüssellänge „sicher" wählen.
Bis zum Jahr 2005 wird geschätzt, daß Schlüssellängen bis 2048 Bit bei asymmetrischen Schlüsseln als sicher anzunehmen sind, bis zum Jahr 2015 Längen bis zu 4096 Bit. Genaue Vorhersagen kann man jedoch nicht treffen, da sich gerade die Entwicklungen im technischen Bereich sehr schlecht abschätzen lassen. Dennoch kann man im Moment bei Digitalen Signaturverfahren davon ausgehen, daß man mit Schlüssellängen ab 1024 Bit sehr gute Sicherheit erwarten kann.

Teil II

Digitale Signaturen in der Praxis

Kapitel 6

Zertifizierung öffentlicher Schlüssel

Die Sicherheit Digitaler Signaturverfahren hängt, wie auch alle anderen kryptographischen Verfahren, von der Geheimhaltung verwendeter Schlüssel ab. Die Frage, die sich hierbei stellt ist, wie die bei Digitalen Signaturverfahren verwendeten Schlüssel, verwaltet werden können, um deren Generierung und Verteilung sicher gestalten zu können. Zudem sollte der in *Kapitel 5* geschilderte Impersonationsangriff so weit wie möglich vermieden werden können. Dies kann durch sog. *Zertifizierung* von Schlüsseln erfolgen. Hierbei wird an den eigentlichen Schlüssel eine Art Stempel: das *Zertifikat* einer vertrauenswürdigen Stelle (Person, Amt) so angefügt, daß der Empfänger des Schlüssels über die Herkunft eine Bestätigung „in Händen" hält. Der „Stempel der Zertifizierungsstelle" besteht meist aus einem geheimen Schlüssel, den nur die Zertifizierungsstelle selbst kennt. Mit diesem Schlüssel und einem Public-Key-Verfahren, kann diese Instanz einen Schlüssel so zertifizieren, daß Empfänger eines derartigen zertifizierten Schlüssels mittels des zugehörigen öffentlichen Schlüssels, den Schlüssel überprüfen können und somit das Zertifikat verifizieren. Ist das Zertifikat echt, so erhält man dadurch den gewünschten Schlüssel.
Die Schlüsselverwaltung kann sich hierbei an unterschiedlichen Infrastrukturen orientieren. Zwei gängige Methoden sind hierbei das „Web of Trust" und eine sog. „hierarchische Zertifizierungsinfrastruktur".

6.1 Das „Web of Trust" als Zertifizierungsinfrastruktur

Zum einen besteht die Möglichkeit ein sog. *Web of Trust*, wortwörtlich übersetzt „ein Netz des Vertrauens" aufzubauen, um Schlüssel sicher verwalten zu können. Ein Kommunikationsteilnehmer „Bob" möchte mit „Alice" Kontakt aufnehmen. Dazu müssen Alice und Bob Schlüssel auf sicherem Wege austauschen, sei es nun,

um später mit einem geheimen Sitzungsschlüssel sicher kommunizieren zu können oder mittels eines Public-Key-Verfahrens Nachrichten auszutauschen. Letzteres benötigt die Übermittlung des (meist) von den Teilnehmern selbst generierten öffentlichen Schlüssels an den Kommunikationspartner. Hierbei ist es notwendig, daß sich Alice und Bob gegenseitig vertrauen können. Der jeweils andere muß sich sicher sein können, daß der empfangene Schlüssel authentisch ist, d.h. daß er auch von der Person stammt, die sie vorgibt zu sein. Ansonsten könnte es zu den gefürchteten Impersonationsangriff kommen (s. *Kapitel 5*). Die wohl vertrauenvollste und sicherste Methode des Schlüsselaustauschs wäre, wenn Alice und Bob sich persönlich kennen würden und die Schlüssel bei einem Treffen „von Angesicht zu Angesicht" (z.B. in Form von Disketten, auf denen der jeweilige Schlüssel gespeichert ist) austauschen könnten. Alice und Bob speichern den übermittelten Schlüssel in einer entsprechenden Datei, um Nachrichten, die mit dem privaten Schlüssel des anderen signiert wurden, mit dem jeweils gespeicherten öffentlichen Schlüssel verifizieren zu können.
Das „Web of Trust" wird dann wie folgt „gesponnen". Möchte ein Bekannter von Bob, nennen wir ihn Tom, ebenfalls mit Alice kommunizieren, so kann Bob für den öffentlichen Schlüssel von Alice bürgen, d.h. Tom garantieren, daß dieser auch von Alice stammt und diesen an Tom weiterleiten. Ebenso bürgt Bob bei Alice für Tom. Tom kann dann seinerseits bei seinen Bekannten für Bob und Alice bürgen usw. Abbildung 6.1) stellt das so entstehende netzwerkartige Vertrauensmodell bildlich dar. Die Zertifizierung des Public Keys eines Teilnehmers erfolgt somit durch andere Teilnehmer, denen bereits vertraut wird.
Der *Vorteil* dieses Schlüsselverwaltungsmodells ist, daß keine übergeordnete Stelle miteinbezogen werden muß, um Kommunikationsschlüssel zu erhalten. Jeder Teilnehmer verwaltet seinen eigenen „Schlüsselbund" mit öffentlichen Schlüsseln der gewünschten Kommunikationspartner. Somit entsteht kein Engpaß wie z.B., wenn mehrere Teilnehmer sich gleichzeitig an eine übergeordnete Schlüsselverwaltungsstelle wenden müßten (s. Abschnitt 6.3).
Der *Nachteil* einer solchen Schlüsselverwaltung ist jedoch, daß man früher oder später die Übersicht verliert, welcher Teilnehmer, welche Schlüssel an seinem Schlüsselbund aufbewahrt. Ist z.B. der Schlüssel eines Teilnehmers nicht mehr gültig, so können die Besitzer des nun veralteten Schlüssels nicht gezielt darüber in Kenntnis gesetzt werden. Die Rücknahme einmal „ausgegebener" Schlüssel erweist sich somit als äußerst aufwendig, da alle Kommunikationsteilnehmer, ob sie nun den besagten Schlüssel kennen oder nicht, über Veränderungen in Kenntnis gesetzt werden müssen. Hinzu kommt, daß es keine Liste gibt, die alle Teilnehmer dieses Netzes aufführt, so daß man sich auch nie sicher sein kann, daß alle Teilnehmer eine derartige „Rückruf-Nachricht" erhalten haben.
Praktische Anwendung findet dieses Schlüsselverwaltungsmodell z.B. beim PGP (Pretty Good Privacy-Verfahren). Dieses wird zunehmend im privaten Nachrichtenaustausch per E-Mail eingesetzt und im folgenden Kapitel zum besseren Verständnis des „Web Of Trust" genauer vorgestellt.

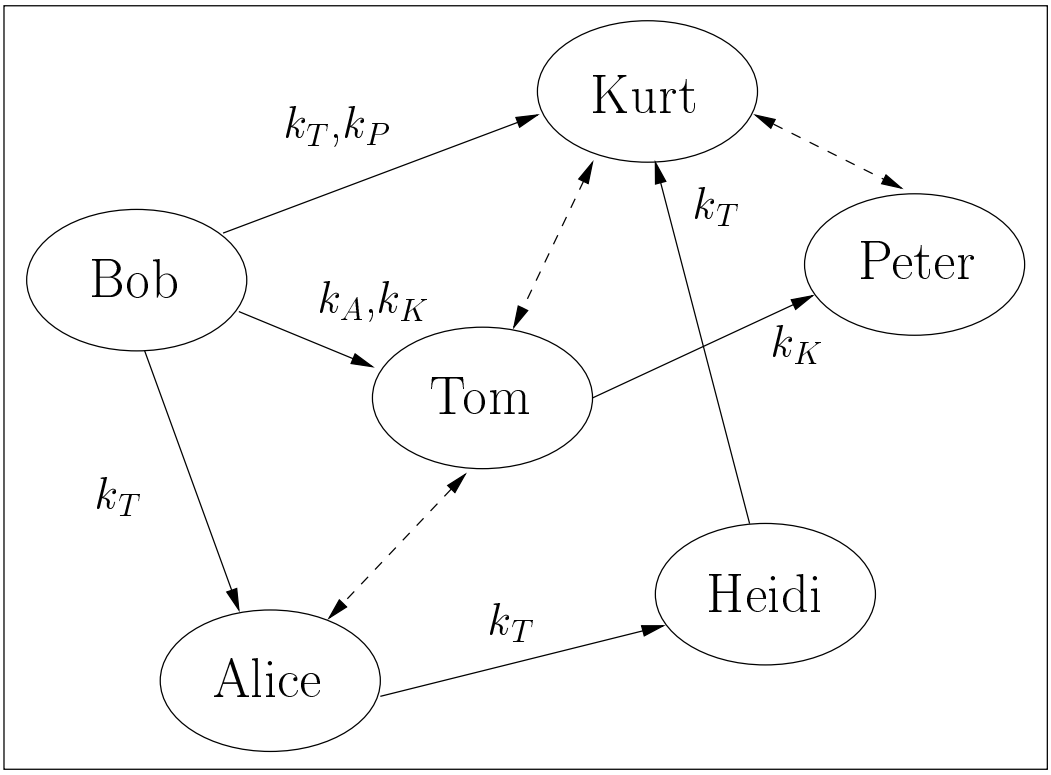


Abbildung 6.1: Die Abbildung veranschaulicht den Kommunikationsaustausch in einem „Web of Trust"
Bob zertifiziert sein Vertrauen zu Alice gegenüber Tom indem er Tom den öffentlichen Schlüssel von Alice zukommen läßt und sein Vertrauen zu Tom gegenüber Alice, indem er Alice Toms öffentlichen Schlüssel übermittelt. Da Tom und Alice Bob absolut vertrauen, können sie anschließend sicher Informationen austauschen (s. $\longleftrightarrow$).
Angenommen Heidi und Bob besitzen gegenüber Kurt nur eingeschränktes Vertrauen, so benötigt Kurt von beiden eine Vertrauensbezeugung, um Tom bzw. dessen öffentlichen Schlüssel Vertrauen schenken zu können. Heidi hat in diesem Fall Toms öffentlichen Schlüssel von Alice erhalten, der sie vollkommen vertraut. Tom kennt Kurts öffentlichen Schlüssel von Bob, so daß die Kommunikation zwischen Tom und Kurt nun „vertrauensvoll" stattfinden kann. (usw.)

6.2 Das „Web of Trust" am Beispiel des PGP

PGP (Pretty Good Privacy) wurde primär von Philip Zimmermann entwickelt. Dies ist ein über das Internet öffentlich zugängliches Verfahren und gleichnamige PC-Software, mit der E-Mails verschlüsselt und signiert werden können. Um dieses Signiersystem nutzen zu können, muß sich der jeweilige Anwender einen privaten und öffentlichen Schlüssel generieren bzw. durch die bereitgestellte Software generieren lassen. Der öffentliche Schlüssel wird dann entweder über einen öffentlich bekannten Server anderen Kommunikationspartnern zugänglich gemacht oder „eigenhändig"/privat an die entsprechenden Kommunikationspartner verteilt. Jeder PGP-Teilnehmer besitzt einen „virtuellen" Schlüsselbund (eine Datei), an den er die öffentlichen Schlüssel anderer Kommunikationsteilnehmer „anhängt" (speichert). Dieser enthält für jeden Eintrag den öffentlichen Schlüssel, die Kennzeichnung des Anwenders dieses Schlüssels (den Namen), die Schlüsselkennzeichnung in Form einer eindeutigen ID (Identifikationsnummer) und Informationen über die Vertrauenswürdigkeit des Schlüsselbesitzers (s.u.).
Benötigt ein Kommunikationsteilnehmer einen Signaturschlüssel, so führt er mittels der PGP-Software folgende *Schlüsselgenerierung* durch:
Der private Schlüssel wird aus einer Passphrase[1] und dem MD5-Algorithmus, s. *Kapitel 7* zu einem 128 Bit-Schlüssel berechnet. Dieser wird mittels des IDEA-Algorithmus (s. Anhang A), verschlüsselt. Der Schlüsselbund für private Schlüssel beinhaltet den verschlüsselten privaten Schlüssel, die Anwenderkennzeichnung des Besitzers und die Schlüsselkennzeichnung des dazugehörigen öffentlichen Schlüssels. Wie anfangs in *Kapitel 2* erwähnt, werden häufig Hybridverfahren verwendet, um die Effizienz eines Verfahrens zu erhöhen. Dies wird auch bei der PGP-Signaturgenerierung angewandt.
Die *Berechnung der Digitalen Signatur* mittels PGP wird dann folgendermaßen durchgeführt:
Zunächst wird das Dokument x erstellt. Dann wird mittels MD5 ein 128 Bit-Hash-Code von x berechnet. Der Unterzeichner bestimmt anschließend den zu verwendenden privaten Schlüssel und gibt eine Passphrase an, mit dem PGP diesen entschlüsseln kann.
PGP verschlüsselt den Hash-Code mittels RSA (s. *Kapitel 7*) (verwendet dazu den entschlüsselten privaten Schlüssel des Unterzeichners) und fügt das Ergebnis an das Dokument an. Die Schlüsselkennzeichung des bei der Verifikation zu verwendenden öffentlichen Schlüssels wird an die Unterschrift angefügt.
Anschließend wird die Nachricht bestehend aus Datei, Signatur und Schlüsselkennzeichnung des öffentlichen Schlüssels mittels einer Komprimierungsfunktion verkleinert, um anschließend zum Schutze der Vertraulichkeit diese mittels des symmetrischen Kryptoverfahrens IDEA verschlüsselt.

[1] Eine Passphrase stellt nicht ein einzelnes Paßwort, sondern einen dementsprechenden ganzen Satz dar.

Der dazugehörige symmetrische Chiffrierschlüssel wird speziell für diese Nachricht mittels eines Zufallsmechanismus erzeugt und mit dem Secret Key des Empfängers und RSA verschlüsselt an die Nachricht angehängt.
Der Verifizierer entschlüsselt zunächst diesen Secret Key mit seinem geheimen RSA-Schlüssel und kann dann die Nachricht mit diesem Schlüssel und mittels des IDEA-Verfahrens entschlüsseln. Dann ermittelt er mittels besagter Schlüsselkennzeichnung den richtigen öffentlichen Schlüssel an seinem Schlüsselbund.
Der Hash-Code wird mit RSA und dem öffentlichen Schlüssel entschlüsselt und mit dem, aus dem Dokument selbst berechneten Hashwert verglichen. Stimmen die beiden Hashwerte überein, so gilt die Unterschrift als verifiziert.
Durch folgende Maßnahmen realisiert das PGP-System zweifache Sicherheit:

- Durch die Verwendung von RSA kann der Verifizierer sicher sein, daß nur der Erzeuger des öffentlichen Verifikations-Schlüssels die Unterschrift mit dem zugehörigen Signaturschlüssel erzeugt haben kann, d.h. dadurch wird besagte Authentiziät erfüllt. Hierbei muß man allerdings beachten, daß zuvor überprüft wurde, ob der Schlüssel auch von besagter Person stammt, nicht von jemanden, der sich nur als besagte Person ausgibt. Dazu muß man besagtes Netz des Vertrauens aufbauen (s.u.).

- Durch die Verwendung von MD5 (s. *Kapitel 7*) kann sich ein Anwender sicher sein, daß ein Unbefugter kein neues Dokument erstellen kann, das zu dem erhaltenen Hashwert paßt.

Das „Web of Trust" bei PGP
Bei Pretty Good Privacy gibt es die Möglichkeit den öffentlichen Schlüssel eines Kommunikationspartners über einen anderen Kommunikationspartner zu beziehen, dem man vertraut und der diesen Schlüssel bereits besitzt.
Eine andere Möglichkeit ergibt sich dadurch, daß jeder PGP-Teilnehmer seinen öffentlichen Schlüssel einem Internet-Server bekannt gibt, über den andere Kommunikationspartner den benötigten Schlüssel abrufen können und ggf. an ihren „persönlichen Schlüsselbund" speichern können. Hierbei ist jedoch nicht garantiert, daß sich jeder gegenüber dem Server als die Person ausgibt, die er vorgibt zu sein. Deshalb empfiehlt es sich, sich zu einem, von einem derartigen Server abgerufenen Schlüssel, einen zusätzlichen Identitätsnachweis zu beschaffen. Dies kann z.B. dadurch geschehen, daß man sich zu einem abgerufenen Schlüssel den zugehörigen MD5-Hashwert als eine Art „Fingerabdruck" desselben berechnet, sich dann z.B. telefonisch mit dem Eigentümer des entsprechenden öffentlichen Schlüssels in Verbindung setzt um auf diese Art und Weise den selbst berechneten Hashwert mit dem des Eigentümers zu vergleichen. Stimmen diese überein und kennt man dazu den Eigentümer dieses Schlüssels persönlich, so kann man sich über die Herkunft des Schlüssels relativ sicher sein. Um den Ursprung eines Schlüssels sicherstellen zu können, gibt es zudem die Möglichkeit, Zertifikate

Bobs Public Key	Bobs ID	Bob@u.de	Signatur von Bob	Signatur von Alice

Abbildung 6.2: Aufbau eines PGP-Zertifikats

zu verwenden, die zu einem Schlüssel eine Garantie darstellen, zu welchem Eigentümer dieser gehört. Ein Zertifikat bei PGP sieht folgendermaßen aus
(s. Abb. 6.2): Zu dem Schlüssel selbst gehört besagte Schlüsselkennzeichnung „Bobs ID", die aus dem Schlüssel und einem Zeitvermerk bestimmt wird, anhand derer man den Schlüssel „Bobs Public Key" an einem Schlüsselbund identifizieren kann. Zu diesem Schlüssel gehört desweiteren eine Anwenderkennzeichnung, die der Schlüsseleigentümer (Bob) an seinen Schlüssel angefügt hat, die die E-Mail-Adresse angibt, die Bob verwendet. Zu dieser Anwenderkennzeichnung gehören zwei Unterschriften, eine von Bob selbst und eine von Alice. Alice nimmt ebenfalls an PGP teil und hat zuvor ebenfalls einen privaten und öffentlichen Schlüssel generiert.
Dadurch, daß beide (Bob und Alice) Bobs Schlüssel signieren, können andere Kommunikationsteilnehmer mittels der jeweils öffentlichen Schlüssel die Signaturen verifizieren. Ergeben dabei beide Verfikationen denselben Hashwert, so kann man davon ausgehen, daß der öffentliche Schlüssel von Bob echt ist, sofern man Alice vertrauen kann.
Die Unterschrift von Alice an Bobs Schlüssel gibt an, daß sie glaubt, daß besagter Schlüssel von Bob stammt, daß die zugehörige Anwenderkennzeichnung von Bob und er im Besitz des zugehörigen privaten Schlüssels ist.
Dadurch, daß Bob sein Vertrauen gegenüber Alice und ihrem Schlüssel bezeugt, vertraut auch sein Bekannter Tom dem Schlüssel von Alice, indem er sich auf sein Urteilsvermögen verläßt.
In PGP gibt es die Möglichkeit den Grad des Vertrauens gegenüber einer anderen Person in Abstufungen zu bezeugen:
Jeder der selbst einen Schlüssel generiert besitzt gegenüber PGP volles Vertrauen. D.h. eine dementsprechende Unterschrift signalisiert vollstes Vertrauen.
Von einem eingeschränkten Vertrauen spricht man bei PGP, wenn ein Benutzer die Frage nach der Vertrauenswürdigkeit eines bestimmten Schlüsseleigentümers, mit „meistens" beantworten würde. Diesem Eigentümer wird sodann „eingeschränktes" Vertrauen zugesprochen.
Die Echtheit einer Anwenderkennzeichnung/Schlüsselkombination wird von PGP dann akzeptiert, wenn zwei „eingeschränkt" vertrauenswürdige Personen oder eine voll vertrauenswürdige Person, diese unterschrieben haben (s. dazu Abb. 6.1). (Diese Parameter sind allerdings in PGP auch veränderbar, z.B. auf zwei voll vertrauenswürdige Personen etc.).

Nun als Beispiel ein Text, der mit PGP 5.0 RSA Key signiert wurde:
—BEGIN PGP SIGNED MESSAGE—
Hallo PGP Freunde,
ich hoffe, die Benutzung von PGP wird sich im ganzen Netz verbreiten !
Ciao
Test
—BEGIN PGP SIGNATURE—
Version: PGP for Personal Privacy 5.0
Charset: noconv
iQEVAwUBM+w3Yerr0KnZUc+tAQFJvQf/deOXYZ7DkxxohccxN+Ob9
VO5/FtCE9E8o1FQwsbYvmRJLoPmZJrejF+PodQt3NFFrqjCTdm8+S
y5SSEKA4BkGZDvxReuAyIA3kcckSfyHIWqoTbSdXtkpXm0oA4cuIL
5Wu+YdVYd6m1aAvtbDWI62tDFxuXpt1Jpc4N/gUxT8qdIXwYZ+s5i
oOhYmpGHtUK+f+XopwObUL8r7/GiL1jRmqL5fi5GcijzEdcGuJGIR260d2
Xveu3V2QjpUjYOoXi6QYzEsauRMUpdqgHV++mcm273Q==
=67zB
—END PGP SIGNATURE—

Der *Vorteil* bei der Verwendung von PGP ist, daß diese Software über das Internet frei zugänglich ist, d.h. jeder diese Software ohne Lizenzgebühren verwenden kann.
Der *Nachteil* dieses Systems ist, wie bereits erwähnt, daß man sich über die Identität des andern nicht immer hundertprozentig sicher sein kann, d.h. daß vor einem Impersonationsangriff wie unter *Kapitel 5* beschrieben wurde, nicht vollkommen geschützt ist. Daher eignet sich PGP nur für private Zwecke. Für geschäftliche, absolut verbindliche Transaktionen sollte immer eine Instanz mit involviert sein, die die Authentizität eines Transaktionsteilnehmers eindeutig beweisen kann. Zudem ist es schwer, einmal veröffentlichte Schlüssel wieder zurückzunehmen, da diese evtl. bereits über Dritte oder Vierte usw. weitergegeben worden sind. Oft kann ein Herausgeber eines Schlüssels gar nicht mehr nachvollziehen, wer nun seinen öffentlichen Schlüssel am Schlüsselbund trägt und wer nicht (s.o.).

6.3 Hierarchische Zertifizierungsinfrastruktur

Eine weitere Möglichkeit des Schlüsselmanagements besteht darin, die Schlüsselverwaltung hierarchisch zu gestalten und durch mehrere Instanzen (z.B. TLCA, PCA, CA, RA und TN) zu strukturieren.
Man erhält somit ein baumartiges Vertrauensmodell, s. Abb 6.3 mit einer Wurzelinstanz, die allen CAs (Certifications Authorities) übergeordnet ist, d.h. über die man zu allen zertifizierten Schlüsseln Zugang findet. Eine CA ist dafür zuständig, Schlüssel der TN (Teilnehmer) zu zertifizieren und zertifizierte Schlüssel anderen autorisierten Teilnehmern zu übermitteln, damit diese sich nicht erst an den

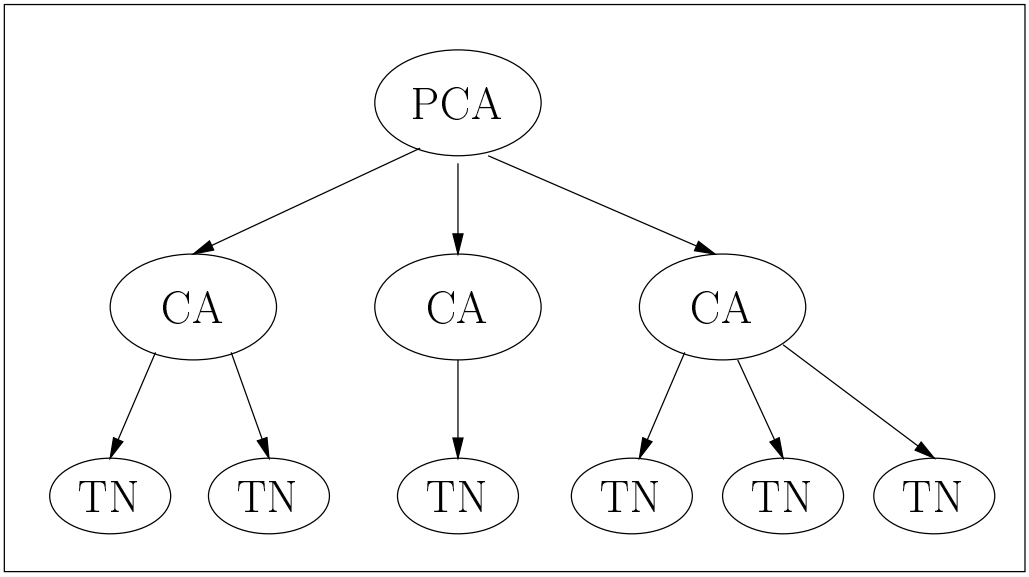


Abbildung 6.3: Diese Abbildung veranschaulicht eine hierarchische Zertifizierungsinfrastruktur, bestehend aus einer *Policy Certification Authority* (PCA), aus mehreren *Certification Authorities* (CAs) und aus *Teilnehmern* (TN) an dieser Schlüsselverteilungsinfrastruktur.

gewünschten anderen TN wenden müssen, um dessen Schlüssel zu erhalten, sondern nach Erhalt des Schlüssels von der CA direkt mit der gewünschten Nachrichtenübermittlung beginnen können.
Da diese Art der Schlüsselverwaltung bei SET (*Kapitel 9*) und somit im Bankwesen angewendet wird und zudem durch das BSI (Bundesamt für Sicherheit in der Informationstechnik) Eingang in die Behörden des Bundes findet, wird dieses Modell im folgenden eingehender vorgestellt.

Die *PCA (Policy Certification Authority)* stellt die Wurzel der Zertifizierungsinfrastruktur dar und dient als Grundlage für jeglichen Informationsaustausch. Der öffentliche Schlüssel dieser Instanz ist allgemein bekannt und ermöglicht somit einen sicheren Datenaustausch zwischen dieser und anderen Instanzen. Die PCA ist als solche für die Verteilung und Prüfung von öffentlichen Schlüsseln und für die Verifikation von Digitalen Signaturen verantwortlich. Sie ist als einzige dazu berechtigt, Zertifikate für die CAs auszustellen, wodurch diese erst berechtigt werden, an der Infrastruktur teilzunehmen.
Der PCA untergeordnet sind die sogenannten *CAs (Certification Authorities)*, die - wie bereits erwähnt - nur von der PCA zertifiziert werden können. Die Aufgabe der CAs besteht darin, Schlüssel für Teilnehmer zu generieren und Teilnehmer mit Zertifikaten auszustatten, damit diese an der Kommunikation teilnehmen können.

Zu jeder CA können eine oder mehrere *RAs (Registration Authorities)* gehören, welche den direkten Kontakt zu den Kommunikations-Teilnehmern herstellen. Diese Aufgabe kann aber ggf. auch von der CA selbst übernommen werden. Die RAs dienen den Teilnehmern als Anlaufstelle für Anträge, wie z.B. Ausstellung, Erneuerung oder Sperrung von Zertifikaten. Dies beinhaltet die Prüfung der Echtheit und Zulässigkeit der Anträge, insbesondere die Identitätsprüfung der Teilnehmer und die Ermittlung aller für die Zertifikatsausstellung benötigten Daten. Zudem übernimmt die RA die sichere Ausgabe von Schlüsselinformationen an die Teilnehmer und informiert diese über einzuhaltende Sicherheitsmaßnahmen. Die RAs stellen als solche eine rein organisatorische Komponente dieser Infrastruktur dar.
Die *TN (Teilnehmer)* sind die eigentlichen Nutzer der Zertifizierungsinfrastruktur, indem sie die verteilten Zertifikate und Schlüsselinformationen zum Schlüsselaustausch und zur sicheren Kommunikation mit anderen Teilnehmern benötigen und anwenden. Sie treten (über ihre RA) mit der für sie zuständigen CA in Verbindung und erhalten über diesen Weg Zertifikate und Schlüssel und können das Sperren oder die Ausgabe neuer Schlüssel veranlassen. [BSI97]
Damit die eben beschriebene Infrastruktur auch bei mehreren bestehenden PCAs funktioniert, gibt es die Möglichkeit, nochmals eine übergeordnete *TLCA (Top Level Certification Authority)* einzuführen. Diese fungiert dann als Wurzel der Zertifizierungshierarchie, welche die Interoperabilität innerhalb der Infrastruktur gewährleistet, indem sie wie die PCA einen allgemeingültigen öffentlichen Schlüssel herausgibt, mit dem die Zertifizierungen der PCAs, die von der TLCA durchgeführt wurden, verifiziert werden können.
Sie kann außerdem bei Unstimmigkeiten aktiv werden und als Vermittler ihre Rolle in diesem System übernehmen. Die hier definierte Hierarchie kann je nach Bedarf systemspezifisch gekürzt werden. In einer konkreten Umsetzung kann eine solche Hierarchie statt auf der Ebene der TLCA auch auf PCA- oder CA-Ebene beginnen.

Wichtig dabei ist, daß

- die Hierarchie stets mit einer für alle Teilnehmer gemeinsamen Wurzel beginnt.
- die Rollen der hier definierten Instanzen beibehalten werden, ggf. um die Aufgaben ergänzt, die durch den Wegfall höherer Hierarchieebenen zusätzlich übernommen werden müssen.
- die Hierarchie stets so aufgebaut ist, daß zu jedem Teilnehmer ein eindeutiger Pfad von der Wurzel der Hierarchie bis zu ihm selbst definiert ist.

Am Beispiel des BSI können die Instanzen wie folgt zugeordnet werden:
PCA:= BSI (Bundesamt für Sicherheit in der Informationstechnik)
CA:= Behörde (Bundesverwaltung: 180 Behörden)
RA:= Personalbüro
TN:= Angestellte einer Behörde
Verzeichnisdienste:=X.500 des IVBB (InformationsVerbund Berlin-Bonn)

- Registrierung und Zertifizierung von Certification Authorities:
  Die Einrichtung einer CA muß unter Angabe des gewünschten CA-Namens und des Zuständigkeitsbereichs bei der PCA beantragt werden. Dabei muß darauf geachtet werden, daß kein CA-Name doppelt vergeben wird und die Sicherheitsleitlinien (Security Policy, die z.B. Namenskonventionen, Vorgehen bei Registrierung und Zertifizierung etc. umfaßt) der zu zertifizierenden CA den Sicherheitslinien der PCA entsprechen.

- Schlüsselgenerierung:
  Die Schlüsselgenerierung kann entweder durch die zuständige CA erfolgen und z.B. per Chipkarte dem Teilnehmer zugängig gemacht werden oder der Teilnehmer generiert seinen privaten und öffentlichen Schlüssel selbst und läßt nur seinen öffentlichen Schlüssel durch die CA zertifizieren.

- *Vorgehen zur Schlüsselermittlung:*
  Ein Teilnehmer benötigt den zu diesem Schlüssel gehörenden Zertifizierungspfad, um einen beliebigen öffentlichen Schlüssel erhalten und überprüfen zu können. Der Zertifizierungspfad besteht aus der Folge aller Zertifikate von dem fraglichen Schlüssel bis zur Wurzel der Zertifizierungshierarchie. Für die Überprüfung eines Teilnehmerschlüssels in der oben aufgezeichneten Hierarchie würden folgende Zertifikate benötigt:

  1 das Zertifikat der CA über den öffentlichen Teilnehmerschlüssel

  2 das Zertifikat der PCA über den öffentlichen CA-Schlüssel

  3 das Zertifikat der TLCA über den öffentlichen PCA-Schlüssel

  Der Teilnehmer prüft dann mit Hilfe des ihm bekannten Wurzelschlüssels das Zertifikat der TLCA über den PCA-Schlüssel und erhält somit den öffentlichen Schlüssel der PCA. Mit diesem prüft er das Zertifikat der PCA über den CA-Schlüssel und erhält ebenfalls den öffentlichen Schlüssel der CA. Mit diesem wiederum kann er abschließend das Zertifikat des gesuchten Teilnehmers prüfen und somit sicher in den Besitz des zugehörigen öffentlichen Teilnehmerschlüssels gelangen.
  Die zu überprüfenden Zertifizierungspfade werden umso kürzer, je flacher die Zertifizierungshierarchie ist. Die Überprüfung eines Zertifizierungspfades kann weiter abgekürzt werden, wenn beide Teilnehmer über eine gemeinsame, in der Hierarchie weiter unten liegende Teilbaumwurzel verfügen.

Verfügt der prüfende Teilnehmer neben dem Wurzelschlüssel auch über alle öffentlichen Schlüssel der Zertifizierungsstellen auf seinem eigenen Zertifizierungspfad, so braucht er dann die Überprüfung jeweils nur ab der gemeinsamen Teilbaumwurzel durchzuführen. Da Kommunikationsbeziehungen in einer Hierarchie in der Regel überwiegend lokal, d.h. im Bereich derselben CA oder PCA bleiben, kann hierdurch je nach Situation eine deutliche Rationalisierung erreicht werden [www98b].

- Das Sperren von Zertifikaten:
  Die *Sperrung eines Teilnehmer-Zertifikats* wird entweder durch die CA selbst, durch den Teilnehmer bei der CA oder auf Antrag der zuständigen RA veranlaßt. Die ungültigen Zertifikate werden in eine sogenannte *Sperrliste* eingetragen, die entweder regelmäßig den Teilnehmern übermittelt wird bzw. die in regelmäßigen Abständen für alle Teilnehmer zugänglich veröffentlicht wird. Die Sperrung eines CA-Zertifikats erfolgt analog auf Veranlassung der zuständigen PCA oder durch die CA selbst.

Vorteile dieser hierarchischen Zertifizierungsinfrastruktur:
Ein *Vorteil* dieser Zertifizierungsinfrastruktur liegt darin, daß ein Teilnehmer nicht erst mit einem gewünschten Kommunikationspartner in Kontakt treten muß, um mit ihm kommunizieren zu können, sondern es genügt, wenn er den öffentlichen Schlüssel der Wurzel der Hierarchie (kurz. Wurzelschlüssel) kennt, um mit Hilfe von Zertifikaten, die im Netz verteilt oder mit Nachrichten mitgeschickt werden, jeden anderen öffentlichen Schlüssel eines beliebigen Teilnehmers sicher erhalten und überprüfen zu können.
Ein weiterer Vorteil dieser Zertifizierungshierarchie ist, daß die Rücknahme von Schlüsseln relativ einfach durch einen Eintrag in eine entsprechende Sperrliste erfolgen kann, über die alle Teilnehmer aufgrund der hierarchischen Zertifizierungsinfrastruktur informiert werden können.

Nachteil dieser Zeritifizierungsinfrastruktur:
Der *Nachteil* dieser Zertifizierungshierarchie ergibt sich aus dem Aufbau, daß man den Zertifizierungspfad kennen muß, um eine Authentisierung durchführen zu können. Außerdem ist der Aufwand der Überprüfung größer als beim „Web of Trust", da man u.U. mehr als ein Zertifikat überprüfen muß, bis man den öffentlichen Schlüssel des gewünschten Kommunikationspartners erhält. Jedoch überwiegen die Vorteile, v.a. die Kenntnis aller Teilnehmer und die Möglichkeit, Zertifikate einfacher sperren zu können, diese Nachteile bei weitem, zumindest wenn ein hohes Maß an Sicherheit erzielt werden soll.

6.4 Hierarchische Zertifizierungsinfrastruktur am Beispiel von PEM

Privacy-Enhanced Mail (PEM) (engl. sinngemäß: vertrauliche (elektronische) Post), ist ein Internet-Standard, der vom Internet Architecture Board (IAB) für sichere elektronische Post über das Internet gewählt wurde. Ursprünglich wurde es von der Privacy und Security Research Group (PRSG) entwickelt. Die PEM-Protokolle bieten Verschlüsselung, Authentifizierung, Nachrichtenintegrität und Schlüsselverwaltung und werden in sog. *RFCs (Requests for Comment) genau beschrieben.* Implementierungen zu PEM sind das sog. RIPEM (nach gleichnamigen Entwickler benanntes Riordan's Internet-PEM) und das sog. TIS/PEM, welches von „Trusted Information System" entwickelt wurde.
Zur Verschlüsselung von Nachrichten werden bei PEM symmetrische Kryptosysteme, wie DES oder Triple-DES (s. Anhang A) und zur Gewährleistung von Datenintegrität kryptographische Hashfunktionen, wie z.B. MD2 bzw. MD5 verwendet. Die Schlüsselverwaltung arbeitet mit Public-Key-Zertifikaten, die mittels des RSA-Algorithmus unter Verwendung von 1024-Bit-Schlüsseln nach dem X.509-Standard strukturiert werden. Hierbei weist PEM eine hierarchische Zertifizierungsstrukturauf: Es gibt eine neutrale Zeritifzierungsinstanz, die sog. IPRA-Wurzel (Internet Policy Registration Authority), welche globale Regelungen aufstellt, die für alle untergeordneten Zertifizierungsinstanzen gültig sind. Die PCAs (Policy Certification Authorities), legen ihre eigenen Regelungen zur Registrierung von Benutzern und Organisationen fest und müssen zuvor von der IPRA zertifiziert werden. Ihrerseits können die PCAs die CAs (Certification Authorities) zertifizieren, die dann Benutzer und untergeordnete organisatorische Einheiten (z.B. Abteilungen, Filialen, Tochtergesellschaften etc.) zertifizieren.
Zertifikate werden als X.509-Zertifikat (s.Abb. 6.4) von den CAs ausgestellt. Jeder Benutzer wird unter einem einmaligen Namen aufgeführt und erhält von der CA ein unterschriebenes Zertifikat, welches den *Namen* und den *öffentlichen Schlüssel des Benutzers* enthält. Das Feld *Version* des Zertifikats gibt das Format des Zertifikats an. Die *Seriennummer* ist innerhalb der CA eindeutig. Die *Algorithmenidentifikation* bezeichnet den Algorithmus, der zur Unterzeichnung des Zertifikats verwendet wird und enthält weitere notwendige Informationen. Unter *Aussteller* befindet sich der Name der CA, die das Zertifikat vergeben hat und die Daten unter *Geltungsdauer* beschreiben Anfang und Ende der Gültigkeit des Zertifikats. Das Feld *Betreff* gibt den Namen des Benutzers des Zertifikats an und unter *Public Key des Betreffs* werden der verwendete Algorithmus, benötigte Parameter, sowie der verwendete Public Key aufgeführt. Das letzte Feld des X.509-Zertifikats enthält schließlich die *Signatur der CA*, die dieses Zertifikat ausstellt.

Name des Benutzers
Public Key des Benutzers
Version des Zertifikats
Serien-Nummer
Algorithmen-ID
Aussteller
Geltungsdauer
Betreff
Public Key des Betreffs
Signatur der CA

Abbildung 6.4: Aufbau eines X.509-Zertifikats

Auf diese Art und Weise entsteht sodann, wie unter *Abschnitt 6.3* beschrieben, eine Kette von Zertifikaten, welche einen Vertrauenspfad von einem fremden öffentlichen Schlüssel bis zur eigenen Vertrauensinstanz etabliert. Möchte Alice nun mit Bob kommunizieren, so informiert sie sich in der entsprechenden Datenbank nach dem Zertifizierungspfad von Alice zu Bob, sowie über Bobs öffentlichen Schlüssel und kann sodann mit der Kommunikation beginnen. Nachrichten können mittels PEM verschlüsselt und unterzeichnet, aber auch nur unterzeichnet sein, wobei sie immer gekapselt auftreten, d.h. folgendermaßen aufgebaut sind:

—BEGIN PRIVACY-ENHANCED MESSAGE —
Proc-Type: 4, MIC-ONLY
Content-Domain: RFC822
DEK-Info: DES-CBC, BFF...
Originator-Certificate:
ASDASJFLSAJLFDSJKL243Jslfjskdlfjla...

Issuer-Certificate:
jksldew,m.erjiKLJKLSDjkslkdswe9kfasdf...

message
—END PRIVACY-ENHANCED MESSAGE—

Hierbei gibt „Proc-Type" an, auf welche Art die Nachricht bearbeitet wurde (verschlüsselt, signiert etc.), "Content-Domain" legt die Art der Mail-Nachricht fest. „DEK-Info" liefert Informationen über den DEK (Data Exchange Key), den zur Textchiffrierung verwendeteten Verschlüsselungsalgorithmus und zugehörige Parameter, z.B. DES.

Der wesentliche *Vorteil* von PEM wird durch diese Zertifizierungsinfrastruktur bereitgestellt, indem, wie man bereits unter 6.3 erkennen konnte, mittels Zertifikat-Sperrlisten, Zertifikate eindeutig sperren kann. Zudem wird durch die CAs vor der Zertifikatausgabe eine eindeutige Identifizierung des PEM-Teilnehmers durchgeführt, so daß Impersonationsangriffe vereitelt werden können.

Zusammenfassend kann man feststellen, daß beide Schlüsselverwaltungsverfahren durchaus ihre Berechtigung in ihrem jeweiligen Anwendungsfeld besitzen. Während die hierarchische Zertifizierungsinfrastruktur einen höheren Verwaltungsaufwand, allein schon zur Erstellung der Zertifikate aufweist, hat man jedoch bei „Verteilten Zertifikaten" im „Netz des Vertrauens" keine absolute Garantie für die Authentizität des Kommunikationspartners und die Sperrung existierender „Zertifikate" erweist sich als schwierig. Bei beiden Verfahren müssen zudem Sicherheitsvorkehrungen bedacht werden, die das Wiedereinspielen von abgehörten Nachrichten kenntlich machen, s. *Kapitel 5*. Letztendlich liegt gerade im „Web Of Trust" ein großer Teil der Verantwortung beim Benutzer selbst, da dieser die Vertrauenswürdigkeit eines anderen einschätzen muß bzw. bei hierarchischer Zertifizierungsinfrastruktur die Gültigkeit von erhaltenen Zertifikaten kontrollieren muß.

Kapitel 7

Konkrete Verfahren und Funktionen

In *Kapitel 6* wurden zwei prinzipielle Verfahrensweisen zur Schlüsselverwaltung bei Digitalen Signaturen aufgezeigt. Dabei wurde anhand von PGP und PEM deutlich, daß im Zusammenhang von Digitalen Signaturverfahren in der Praxis meist auch Chiffrier- und Hashfunktionen verwendet werden. Die Ursache dieser „Kopplung" von Verfahren liegt darin, daß man sich zur Gewährleistung von Vertraulichkeit nicht auf Signaturverfahren verlassen kann, sondern auf Kryptosysteme zurückgreifen muß. Im folgenden soll nun aufgezeigt werden, in welchem Zusammenhang verwendete Signatur-, Hash- und Verschlüsselungsfunktionen stehen, wie sie im einzelnen aufgebaut sind und welche Ziele sie konkret verfolgen.

7.1 Digitale Signaturen, Hash- und Chiffrierfunktionen

Die Verwendung Digitaler Signaturen zielt darauf ab, ganz gewisse Anforderungen zu erfüllen. Diese sind, wie in *Kapitel 1* erwähnt, Authentizität, Datenintegriät, Fälschungssicherheit und Verbindlichkeit. In den vorherigen Kapiteln wurde es zum Teil schon ersichtlich, wie diese Anforderungen erfüllt werden können. Abbildung 7.1 verfolgt die Absicht, das Zusammenspiel unterschiedlicher Funktionen, wie sie in praktischen Fällen auftreten, und den jeweiligen Verwendungszweck aufzuzeigen. Eine Transformationsfunktion T stellt je nach Digitalem Signaturverfahren entweder eine Redundanzfunktion R (bei Digitalen Signaturen mit Dokumentenwiederherstellung) oder eine Einweghashfunktion H (bei Digitalen Signaturen als Anhang, s. *Kapitel 4*) dar. Die Verwendung einer Redundanzfunktion zielt, wie bereits unter *Kapitel 5* geschildert wurde, auf erhöhte Sicherheit gegenüber nicht-selektiven Angriffen ab, wie auch die Verwendung von Hashfunktionen, wobei diese zudem den Schutz der Datenintegrität signierter Dokumente garantieren.

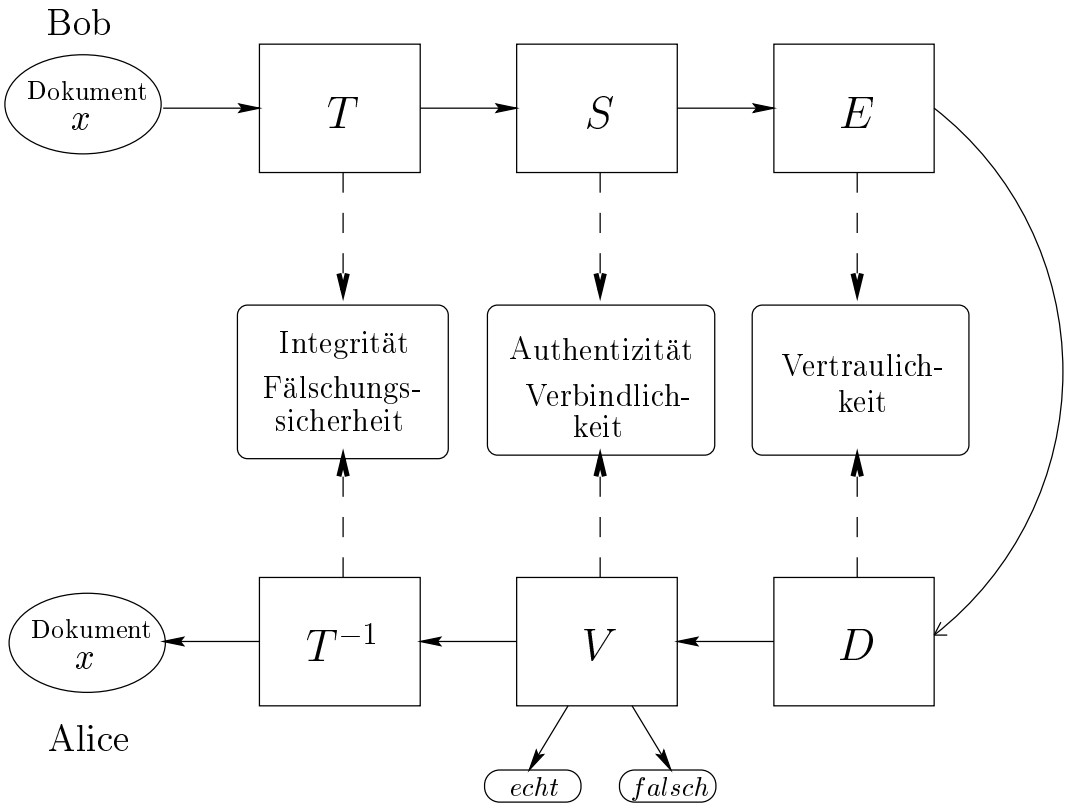


Abbildung 7.1: Übersicht über das Zusammenspiel unterschiedlicher Funktionstypen und ihrem Verwendungszweck

Nach Anwendung einer dieser Funktionstypen wird sodann der entsprechende Wert ($R(x)$ oder $H(x)$) der Signaturfunktion S unter Verwendung des entsprechenden Signaturschlüssels unterworfen, wobei dies zur Authentikation des Unterzeichners beim Verifizierer dient und zugleich Fälschungssicherheit bewirkt. Zudem erreicht man durch die Verwendung von Signaturverfahren besagte Verbindlichkeit, d.h. man kann durch den Verifikationsalgorithmus jederzeit auch gegenüber „Dritten" nachweisen, von wem das Dokument signiert wurde.
Um Vertraulichkeit bei der Übertragung gewährleisten zu können, verwendet man Kryptosysteme, wie es in *Kapitel 2* bereits geschildert wurde. Da symmetrische Kryptosysteme wesentlich schneller als asymmetrische sind, dafür aber das Sicherheitsrisiko des geheimen Schlüsselaustausches mit sich bringen, werden diese Verfahren meist gekoppelt. Der geheime symmetrische Schlüssel, meist auch Sitzungsschlüssel genannt (da er meist für jeden Kommunikationsaustausch neu generiert wird) , wird mittels eines Public-Key-Kryptosystems verschlüsselt und auf diese Weise nur dem gewünschten Kommunikationspartner zugänglich gemacht. Die Verschlüsselung durch ein Kryptosystem garantiert besagte Vertraulichkeit und damit eng verbundene Fälschungssicherheit.
Mittels der Verifikationsfunktion V wird sodann die Authentizität des erhaltenen Dokuments (nach Entschlüsselung) nachgewiesen, wobei hier im Falle Digitaler Signaturen mit Dokumentenwiederherstellung, wie in *Kapitel 5* geschildert, nochmals einer Transformation: der Umkehrfunktion der Redundanzfunktion, unterzogen werden muß, um das signierte Dokument in Händen zu halten. Bei Digitalen Signaturen als Anhang stellt die Transformationsfunktion wieder die Hashfunktion dar. Diese wird auf das übermittelte Dokument angewandt und wie in *Kapitel 4* geschildert mit dem mitgesandten Hashwert verglichen.
Gängige Algorithmen die zur Signaturbildung verwendet werden, sind der schon öfters erwähnte RSA-Signaturalgorithmus, das ElGamal-Signaturverfahren und die MD5-Hashfunktion. Die beiden letzteren finden beim sog. DSS (Digital Signature Standard) Anwendung. Was diese Funktionen als Einweg- bzw. Trapdoorfunktionen auszeichnet (s. *Kapitel 2* und 3: Anforderungen an Signaturfunktionen) wird in folgenden Abschnitten vorgestellt.

7.2 Grundlegende mathematische Begriffe

Bevor nun die Algorithmen im einzelnen beschrieben werden, sollten noch ein paar grundlegende mathematische Begriffe angesprochen werden.
In der Kryptographie vermeidet man das Rechnen mit der Menge der natürlichen Zahlen N, da diese aus unendlich vielen Zahlen besteht, die auf dem Rechner aufgrund begrenzter Speicherkapazitäten auch nur begrenzt [z.B. Werte von 0 bis $2^{16}-1$] darstellbar sind. Stattdessen wird mit sog. Restklassen gearbeitet, die im folgenden definiert werden.

Definition: *Restklassen*
Seien $a, b \in Z, m \geq 1$. a und b heißen *kongruent modulo* m, kurz: $a \equiv b \ (mod\ m)$, falls m die Differenz $a-b$ teilt. Durch diese Äquivalenzrelation wird Z in m *Äquivalenzklassen/ Restklassen* unterteilt:
$[a]_m = \{b \in Z \mid a \equiv b \ (mod\ m)\},\ a = 0, ..., m-1$
Auf dem *Restklassenring* $Z_m = \{[0]_m, ..., [m-1]_m\}$ sind Addition und Multiplikation wie folgt definiert:
$[a]_m + [b]_m = [a+b]_m$
$[a]_m \ \star [b]_m = \ [ab]_m$, wobei $[1]_m$ das Einselement von Z_m darstellt.

Definition: *Multiplikatives Inverses*
Sei $a \in Z_m$. Das Multiplikative Inverse von a *modulo* m ist ein Element $x \in Z_m$, mit $ax \equiv 1 \ (mod\ m)$. Existiert ein derartiges x, dann ist es eindeutig *modulo* m und a heißt invertierbar.
Korollar: Zu einem $a \in Z_m$ existiert ein *multiplikatives Inverses* genau dann und nur dann, wenn $ggT(a, m) = 1$.

Die *invertierbaren Restklassen* $Z_m^* = \{[a]_m \mid ggT(a, m) = 1\}$ werden zur *multiplikativen Gruppe* zusammengefaßt.
Z_m^* ist die Menge aller Reste, die teilerfremd zu m sind. Jedes Element in Z_m^* besitzt ein *multiplikatives Inverses*, welches auch in Z_m^* liegt. Um das jeweilige multiplikative Inverse zu berechnen, kann man den *Erweiterten Euklidischen Algorithmus* (s. [Sti95], S.116f.) verwenden.

Eine wichtige Rolle spielt häufig die *Eulersche* φ*-Funktion*, da sie in vielen Algorithmen, z.B. dem RSA-Algorithmus, Anwendung findet:
Sie entspricht der Mächtigkeit der Menge Z_m^*:
$\varphi:\ N \to N$ mit $\varphi(m) = \parallel Z_m^* \parallel = \parallel \{a \mid 1 \leq a \leq m,\ ggT(a, m) = 1\} \parallel$

Satz von Euler-Fermat:
Für alle $a \in Z_n^*$ gilt: $a^{\varphi(n)} \equiv 1 \ (mod\ n)$, wobei $\varphi(n) = (p-1)(q-1)$;
$p, q\ prim;\ n = pq$ (Beweis dazu, s. [Kö94],S. 36)

Satz von Fermat:
Für alle Primzahlen p und alle Elemente $a \in Z_p^*$ gilt: $a^{p-1} \equiv 1 (mod\ p)$.

Definition: *Erzeuger*
Eine Zahl g heißt *Erzeuger* von Z_m^* oder auch *Primitivwurzel modulo* m, falls $Z_m^* = \{1, g, g^2, g^3, ...g^{\varphi(m)-1}\}$.
Die *Ordnung von g modulo m* ist $ord_m g = min\{e \geq 1 \mid g^e \equiv 1\ (mod\ m)\}$, wobei $m \geq 1$ und $ggT(g, m) = 1$.
g ist genau dann Erzeuger von Z_m^*, wenn $ord_m(g) = \varphi(m)$.

Für Berechnungen mit Restklassen, wird sehr oft, wie man im folgenden noch sehen wird, der folgende Satz benötigt:

Der *Chinesische Restsatz:*
Sind $m_1, ..., m_k$ paarweise teilerfremd und ist
$ggT(a_1, m_1) = ... = ggT(a_k, m_k) = 1$, so hat das System $a_i x \equiv b_i\ (mod\ m_i)$,
$i = 1, ..., k$ genau eine Lösung *modulo* m $(m = m_1 * ... * m_k)$.
Beweis: [Kö94], S. 37.

Lemma *zum Chinesischen Restsatz:*
Falls $x \equiv y\ (mod\ m_i), i = 1, ..., k$, so folgt $x \equiv y\ (mod\ m)$,
wobei $m = kgV(m_1, ...m_k)$. *Beweis:* s. [Kö94], S.37.

7.3 Der RSA-Algorithmus

Der RSA-Algorithmus ist ein nach Rivest, Shamir und Adleman benannter Algorithmus, der auf dem Faktorisierungsproblem basiert und als Trapdoor-Funktion (Def. s. *Kapitel 2*) die *modulare Potenzfunktion*: $f(x) = x^e\ mod\ n$ verwendet, wobei $n = pq$, $p, q\ prim$ und $ggT(e, \varphi(n)) = 1$; $\varphi(n) = (p-1)(q-1)$. Diese zeichnet sich dadurch aus, daß sie durch *Wiederholtes Quadrieren und Multiplizieren*[1] relativ einfach zu berechnen ist, deren Umkehrfunktion jedoch vermutlich nicht, da das hierzu benötigte Multiplikative Inverse d zu $e\ (mod\ \varphi(n))$ jedoch nicht so ohne weiteres bestimmt werden kann (s. 7.3.3). Dieser Algorithmus wird sowohl zur Verschlüsselung als auch zur Signaturbildung häufig verwendet und aufgrund dessen nun eingehender erläutert. Zunächst wird das Verschlüsselungsverfahren beschrieben, da dieses sehr leicht in ein entsprechendes Signaturverfahren umgewandelt werden kann.

[1] Potenzen $x^n\ mod\ m$ lassen sich durch *Wiederholtes Quadrieren und Multiplizieren* folgendermaßen effizient berechnen: *input* $x, m, n = \Sigma_{i=0}^{r} a_i 2^i$ $y := 1$; *for* $i := r$ *downto* 0 *do* $y := y^2 x^{a_i}\ mod\ m$; *output* y

7.3.1 Der RSA-Chiffrieralgorithmus

Der RSA-Algorithmus basiert, wie soeben erwähnt auf dem *Faktorisierungsproblem.* Dies baut darauf auf, daß es zwar einfach ist, zwei Primzahlen miteinander zu multiplizieren, im Gegensatz dazu aber sehr aufwendig ist, aus dem Ergebnis die ursprünglichen Primzahlen wieder zu ermitteln. Bei genügend großen Primzahlen (derzeit geht man von mindestens 100-stelligen Primzahlen aus), ist das Problem faktisch unlösbar. Deshalb werden für jeden Teilnehmer zwei große Primzahlen p und q gewählt und daraus das Produkt $n = p * q$ gebildet, welches dem Modulus der Trapdoor-Funktion entspricht. Dies bedeutet zugleich, daß die Kenntnis dieser Primfaktoren die entsprechende Trapdoor-Information darstellt (s. 7.3.3). Zur **Schlüsselgenerierung** muß man dann noch zwei Zahlen e und d folgendermaßen bestimmen:
Wähle eine kleine, ungerade Integerzahl e, e *relativ prim* zu $\varphi(n)$, d.h. $e \in Z^*_{\varphi(n)}$, $\varphi(n) = (p-1)(q-1)$
Berechne d als das *Multiplikative Inverse* zu e *modulo* $\varphi(n)$, so daß
$ed \equiv 1 \ (mod\ \varphi(n))$ und $1 \leq d \leq \varphi(n)$ mittels des *Erweiterten Euklidischen Algorithmus.* (Laut Korollar in 7.2 existiert d und ist eindeutig.)
Jede „Einheit", die den RSA-Algorithmus verwendet, benötigt ein Schlüsselpaar, das jeweils aus folgenden zwei Zahlen besteht:
(e, n) : Öffentlicher Schlüssel (Public Key)
(d, n) : Geheimer Schlüssel (Secret Key)
Zur **Verschlüsselung** wird der Klartext, der in digitaler Darstellung aus einer Folge von Nullen und Einsen besteht, in $n_i,\ i \in N$ Blöcke bestimmter Länge (z.B. 64 Bit) geteilt und jeder Block als binäre Zahl interpretiert. Wenn x eine solche Zahl ist, ergibt sich der zugehörige verschlüsselte Textblock mit der folgenden mathematischen Operation: $c = x^e \ mod\ n$.
Die Zahl c stellt den verschlüsselten Textblock dar. Entschlüsselt wird er unter Verwendung der Geheimzahl d mit der Operation: $x = c^d \ mod\ n$.
Diese Umkehrung liefert wieder den Klartext, wie der Satz von Euler-Fermat aus der Zahlentheorie beweist:

Satz von Euler-Fermat:
Für alle $a \in Z^*_n$ gilt: $a^{\varphi(n)} \equiv 1 \ (mod\ n)$, wobei $\varphi(n) = (p-1)(q-1)$;
$p, q\ prim;\ n = pq$ (s. auch 7.2)

Damit gilt: $c^d \equiv (x^e)^d \equiv x \ mod\ n,\ \forall x \in Z^*_n$
Beweis für die Richtigkeit der Verifikationgleichung:
Laut Vor. gilt: $ed \equiv 1 \ (mod\ \varphi(n)) \Rightarrow ed = k\varphi(n) + 1, k \in N$;
Sei $x \in Z_n$;

- 1.Fall: $ggT(x, n) = 1$.
  Mit dem *Satz von Euler-Fermat* (s.o.) folgt unmittelbar:
  $x^e d \equiv x^{k\varphi(n)+1} \equiv (x^{\varphi(n)})^k x \equiv x \ mod\ n$, da $x^{\varphi(n)} \equiv 1 \ mod\ n$, falls $x \in Z^*_n$.

- 2.Fall: $ggT(x,n) \neq 1$, d.h. p oder q sind Teiler von x.
  - Ann.: $p \mid x \Rightarrow x = lp, l \in N, l \leq q$
    $\Longrightarrow\ x^e d = (lp)^{k\varphi(n)+1} = (lp)^{k(p-1)(q-1)} lp = lp((lp)^{(q-1)})^{k(p-1)} \equiv$
    $lp \equiv x\ (mod\ q)$, da $(lp)^{(q-1)} \equiv 1\ (mod\ q)$ (Euler-Fermat).
    $\Longrightarrow$ Da p/x gilt weiter: $x^e d = (lp)^{ed} \equiv 0 \equiv x\ (mod\ p)$
  - Ann.: q/x gilt analog.
- Insgesamt ergibt sich aus folgendem *Lemma zum Chinesischen Restsatz*:
  Falls $x \equiv y\ (mod\ m_i), i = 1, ..., k$, so folgt $x \equiv y\ (mod\ m)$,
  wobei $m = kgV(m_1, ..., m_k). \Rightarrow\ x^{ed} \equiv x\ (mod\ pq)$, q.e.d. ([Th.92], S.825).

Wie bereits in *Kapitel 2* ersichtlich wurde, beruht der *Vorteil* dieses Verfahrens darauf, daß die geheimen Schlüssel nicht an andere übermittelt werden müssen. So kann Bob ein Dokument für Alice verschlüsseln ohne den privaten Schlüssel von Alice zu kennen und Alice (d.h. nur sie) kann das Dokument lesen.
Der *Nachteil* dieses Verfahrens liegt jedoch bei der Berechnungsgeschwindigkeit. Es lohnt sich meist nur, kleine Dokumenten mit RSA zu verschlüsseln, da ansonsten die Ver- und Entschlüsselungszeiten zu groß werden ([AB95], S.19).

7.3.2 Der RSA-Signaturalgorithmus

Wie bereits in *Kapitel 2* geschildert wurde, kann man aus einem derartigen Public-Key-Verfahren ein Signaturverfahren generieren. Entsprechend der dort geschilderten „Umwandlung" wird die Entschlüsselungsfunktion zur Signaturfunktion:
$S : x \rightarrow x^d\ mod\ n$,
die Verschlüsselungsfunktion zur Verifikationsfunktion:
$V : z^e\ mod\ n \rightarrow \{echt, falsch\}$, wobei z die Signatur darstellt.
Der geheime Schlüssel (d,n) wird zum Signaturschlüssel und der öffentliche Schlüssel (e,n) wird zum Verifikationsschlüssel.
Das so entstehende Signaturverfahren stellt an sich eines mit Dokumentenwiederherstellung dar. Da jedoch mittels RSA nur Blöcke fester Länge signiert werden können und man nicht-selektive Angriffe durch Blockvertauschung (s. dazu auch *Kapitel 4*) vermeiden möchte, wendet man in der Praxis meist zuvor auf das Dokument eine Einweghashfunktion an, um dann den Hashwert (fester Länge), s. *Kapitel 4* zu signieren. Dies entspricht der in *Kapitel 5* beschriebenen Umwandlung eines Signaturverfahrens mit Dokumentenwiederherstellung in Digitale Signaturen als Anhang.
Angenommen es liegt nun ein Dokument x fester Länge vor, d.h. $x \in [0,1]^n$, so berechnet Bob mittels des RSA-Signaturverfahrens die Signatur: $z = x^{d_B}\ mod\ n$ und Alice kann diese mittels der Berechnung: $z^{e_B}\ mod\ n = (x^{d_B})^{e_B}\ mod\ n = x$ auf Echtheit überprüfen. Dies gilt laut obigen Beweis mittels Euler-Fermat.[2]

[2]Hierzu ist zu bemerken, daß die Signatur letztendlich dadurch verifiziert ist, daß, im Falle von Dokumentenwiederherstellung entweder x einen sinnvollen Text ergibt bzw. bei Anwendung

7.3.3 Sicherheit des RSA-Algorithmus

Wie in *Kapitel 2* bereits erörtert, müssen Public-Key-Verfahren Trapdoor-Funktionen darstellen, um ein Mindestmaß an Sicherheit gewährleisten zu können. Die hierbei verwendete *modulare Potenzfunktion* baut, wie gesagt, auf dem Faktorisierungsproblem auf. Dies bedeutet, daß die Sicherheit des Verfahrens von der Geheimhaltung der Primfaktoren abhängt und somit folgende Trapdoor-Eigenschaft erfüllt sein muß:
Trapdoor-Eigenschaft des RSA-Verfahrens:
„Kennt man nur die Zahl n, aber nicht die Faktoren p und q bzw. die Zahl $\varphi(n) = (p-1)(q-1)$, so kann man d nicht aus e berechnen."
Sind jedoch p und q bekannte Faktoren, so läßt sich aus dem öffentlichen Schlüssel (e, n) der geheime Schlüssel (d, n) berechnen, mit dem daraufhin beliebige Nachrichten gefälscht werden können, denn laut Vor. gilt:
$ed \equiv 1 \ mod \ (p-1)(q-1)$ Daraus kann man das *Multiplikative Inverse* zu e, nämlich d, mittels des Erweiterten Euklidischen Algorithmus (s. 7.2) relativ einfach berechnen.
Anm.: Eine bisher noch nicht geklärte Frage, die sich hierbei stellt, ist, ob das Kryptosystem evtl. auf ganz andere Weise gebrochen werden könnte, als über die Kenntnis von p und q? Derzeit ist allerdings kein anderer Weg als über die Primfaktorenzerlegung bekannt.

Eine Möglichkeit, die gesuchten Primfaktoren zu ermitteln, besteht, wenn man n und $\varphi(n)$ kennt. Die Berechnung für die Bestimmung der Primfaktoren p und q lautet dann wie folgt:
$\varphi(n) = (p-1)(q-1) = (p-1)(n/p-1)$, da $pq = n, \ p > q$
Dies ergibt durch Umformungen eine quadratische Gleichung, die nach p aufgelöst werden kann, woraus sich dann auch q bestimmen läßt:
$\varphi(n) = n - p - \frac{n}{p} + 1 \Rightarrow p^2 - (n + 1 - \varphi(n))\ p + n = 0$
Sei $b := n + 1 - \varphi(n) \Rightarrow p = \frac{k}{2} + \sqrt{(\frac{b}{2})^2 - n}; \ \ q := n/p$
Daraus kann man dann wie oben beschrieben, den geheimen Schlüssel berechnen.

Außerdem sollte das *Kleinste gemeinsame Vielfache* $k = kgV(p-1, q-1)$ möglichst groß sein, da sonst durch Raten oder Ausprobieren ein zu d äquivalentes d^* gefunden werden kann, mit dem man anstelle von d die Dokumente entschlüsseln kann. Denn es gilt, d ist äquivalent zu d^*, falls die Differenz $d - d^*$ ein Vielfaches des kgV k darstellt. Dies zeigen folgende Äquivalenzumformungen:
$\forall x: \ x^{ed^*} \equiv x \ (mod \ n)$
$\Leftrightarrow \forall x: x^{ed^*} \equiv x \ (mod \ p) \ \wedge \ \forall x: \forall x^{ed^*} \equiv x \ (mod \ q)$

---

einer Redundanzfunktion im Bildbereich dieser Funktion liegt. Im Falle Digitaler Signaturen als Anhang, stellt x den entsprechenden Hashwert des Dokuments dar und muß wie in *Kapitel 4* beschrieben mit dem selbstberechneten Hashwert verglichen werden.

$\Leftrightarrow (p-1)/(ed^*-1) \ \wedge \ (q-1)/(ed^*-1)$ (*Satz von Gauß*[3]).
$\Leftrightarrow ed^* \equiv 1 \ (mod \ p-1) \ \wedge \ ed^* \equiv 1 \ (mod \ q-1)$
$\Leftrightarrow ed^* \equiv 1 \ (mod \ k)$
$\Leftrightarrow d^* \equiv d \ (mod \ k)$
Da nun $(p-1)(q-1) = kgV(p-1, q-1)ggT(p-1)(q-1)$, sollte der $ggT(p-1)(q-1)$ folglich relativ klein sein, d.h. $(p-1)$ und $(q-1)$ müssen große, nicht gemeinsam vorkommende Primfaktoren enthalten.
Liegen die Primfaktoren p und q zu nahe beieinander, so können diese durch die *Differenz der Quadrate* faktorisiert werden (s. [Kö94], S.59).
Zu bedenken ist außerdem, daß durch die Verwendung des gleichen RSA-Moduls $n = pq$, jeder die Dokumente des anderen lesen kann ([AB95], S.21).

Was bedeutet diese Trapdoor-Eigenschaft jedoch für das RSA-Signaturverfahren?

Die Geheimhaltung der Primfaktoren stellt auch beim RSA-Signaturverfahren eine grundlegende Anforderung für die Sicherheit dar. Denn analog zu obigen Berechnungen kann man für bekanntes p und q den geheimen Signaturschlüssel berechnen und somit die Unterschrift des Schlüsseleigentümers exakt fälschen.
Jedoch ist bei Digitalen Signaturverfahren noch eine „stärkere" Eigenschaft gefordert (s. *Kapitel 2*): Es muß für einen Gegner zudem unmöglich sein ohne Kenntnis des Signaturschlüssels eine Signatur z zu finden, so daß der Verifikationsalgorithmus V diese als *echt* anerkennt. (*Fälschungssicherheit*)
Wie man bereits in *Kapitel 5* sehen konnte, kann man dies nicht hundertprozentig ausschließen: Ein Gegner sucht sich zufällig eine Signatur und überprüft diese mittels des öffentlich bekannten Verifikationsalgorithmus auf ihre Gültigkeit. Wie dort beschrieben, kann man durch die Verwendung von Redundanzfunktionen die Erfolgswahrscheinlichkeit dieses nicht-selektiven Angriffs auf ein Minimum reduzieren.
Eine andere Möglichkeit Angriffe auf das RSA-Signaturverfahren zu vereiteln, besteht, wie unter *Kapitel 4* beschrieben, durch die Verwendung von Einweg-Hashfunktion, die es, falls man mittels nicht-selektivem Angriff eine gültige Signatur ermitteln kann, so gut wie unmöglich macht, das zugehörige Dokument ausfindig zu machen.
Zudem besteht beim RSA-Algorithmus aufgrund der Multiplikativität die Möglichkeit, selektive Angriffe auszuführen.
Multiplikativität (des RSA-Algorithmus):
Es gilt: $\forall x, y \in X : S(xy) = S(x)S(y)$, wobei S, den RSA-Signaturalgorithmus darstellt.
Ein Angreifer muß in diesem Falle „nur" zwei Dokumente x, y finden, so daß $xy = m \ mod \ n$ ergibt. Kann er sich zu x und y, die gültigen Signaturen beschaffen, so ist er automatisch im Besitz einer gültigen Signatur zu m.

[3]Der *Satz von Gauß* besagt: Ist p eine Primzahl, so gibt es genau $\varphi(p-1)$ Erzeuger(s. 7.2).

Eine Abwandlung dieses Angriffs stellt die sog. *Moore-Attacke* dar (s. [Mis98]), benannt nach Judy Moore, die diesen Algorithmus ursprünglich als Entschlüsselungsalgorithmus entwickelt hatte, um dann festzustellen, daß er zum Fälschen von Signaturen verwendet werden kann:
1. Sei x das Dokument, zu dem die Signatur z gefälscht werden soll.
2. Sei z_1 ein Zufallswert.
3. Berechne $x_1 = z_1^e \ (mod\ n)$, so daß (x_1, z_1) ein gültiges Dokument-Signaturpaar darstellen.
4. Beschaffe die Signatur z_2 zu x_1x: $z_2 = S(x_1x) = (x_1x)^d \ (mod\ n)$.
5. Berechne sodann $z = z_1^{-1}z_2 = z_1^{-1}(x_1x)^d = z_1^{-1}z_1x^d = x^d \ (mod\ n)$ und erhalte so die gültige Signatur zu dem Dokument x.

Dieses Verfahren bietet im Vergleich zum vorherigen Angriff den *Vorteil*, daß der Angreifer nur eine (anstatt) zwei Original-Signaturen von seinem „Opfer" (d.h. dessen Unterschrift er fälschen möchte) benötigt, nämlich die Signatur zu x_1x.

Eine weitere Angriffsmöglichkeit auf den RSA-Signaturalgorithmus besteht aufgrund seiner *Reziprok-Eigenschaft*. Dies bedeutet, es gilt:
$D(E(x)) = E(D(x)) \ (mod\ n)$, d.h. $(x^e)^d = (x^d)^e \ (mod\ n)$. Ein möglicher Angriff, der auf dieser Eigenschaft basiert, ist der sog. Gordon Angriff.
Der Gordon Angriff (s. [Mis98], S.19) stellt einen nicht-selektiven Angriff dar. Gordon geht hierbei davon aus, daß eine vertrauliche Stelle, direkt den öffentlichen Schlüssel von Alice (e_A, n_A) mit der geheimen Signaturfunktion S unterzeichnet. Zudem werden e_A und n_A separat unterschrieben, so daß $c_1 = S(e_A)$ und $c_2 = S(n_A)$. Ein Zertifikat besteht dann aus: $\{c_1, c_2, e_A, n_A\}$
Gordons Attacke geht neben der Tatsache, daß der RSA-Algorithmus reziprok ist davon aus, daß der Modulus n_A faktorisiert werden kann. Das Faktorisieren wird dadurch erreicht, indem man die Funktion P der vertraulichen Stelle auf einen Zufallswert c_2 anwendet (s.u.).

- Angenommen der Angreifer möchte ein gültiges Zertifikat $\{c_1, c_2, e_A, n_A\}$ generieren.
- Dann wählt der Angreifer zufällig ein c_1 und ein c_2.
- Er berechnet e_A und n_A, indem er P auf c_1 und c_2 anwendet:
  $e_A = P(c_1)$ und $n_A = P(c_2)$.
- Kann n_A nicht faktorisiert werden bzw. besitzt gewisse Primfaktoren mehrfach, dann beginnt der Angreifer nochmals bei b.
  Ansonsten seien $p_1, ..., p_k$ die Primfaktoren zu n_A.
- Der Fälscher kann hierzu $\varphi(n_A)$ berechnen:
  $T = \varphi(n_A) = (p_1 - 1)(p_2 - 1)...(p_k - 1)$ berechnen.

- Durch Verwendung des *Erweiterten Euklidischen Algorithmus*, kann er versuchen d_A zu berechnen, so daß $e_A d_A = 1(mod\ T)$. Er kann dies dann erfolgreich tun, wenn der $ggT(e_A, T) \neq 1$ ist. Gilt dies nicht, so geht der Fälscher zurück zu Schritt b und wählt sich ein neues c_1. Ansonsten ist $\{c_1, c_2, e_A, d_A\}$ ein gültiges Zertifikat und der Fälscher kennt d_A, so daß er mittels dem Satz von Euler-Fermat (s. 7.2) $x = x^{e_A d_A}\ (mod\ m)$ berechnen kann.

Um Angriffsarten, die auf die Multiplikativität des RSA-Algorithmus abzielen, vereiteln zu können, kann man sog. „Padding" anwenden. Hierbei werden „kürzere" Nachrichten durch bestimmte „Padding-Verfahren" auf eine bestimmte Länge gebracht. Dadurch können oben beschriebene algebraische Zusammenhänge, wie z.B. (S(xy)=S(x)S(y)) zerstört werden und derartige Angriffe daher vereitelt werden.
Zudem sollten Redundanz- (s. *Kapitel 5*) bzw. Hashfunktionen (s. *Kapitel 4*) mitverwendet werden, um Attacken, die auf der Reziprok-Eigenschaft aufbauen, vereiteln zu können, indem dadurch die Wahrscheinlichkeit auf Erfolg auf ein Minimum verringert wird.
Anm.: Es gibt allerdings auch Attacken, die trotz verwendeter Hash- oder Redundanzfunktionen, Erfolg haben können. Allerdings ist die Wahrscheinlichkeit extrem gering, daß derartige Angriffe auch durchgeführt werden können, d.h. die dazu benötigten Voraussetzungen auch vorliegen. Dies veranschaulicht folgendes Beispiel: die *Coppersmith-Attacke*. Hierbei konnte eine Schwachstelle einer verwendeten Hashfunktion aufgedeckt werden. Der „Angreifer" konnte zwei zufällige Dokumente finden x, x^*, die verwandte Hashwerte aufwiesen, mit: $h(x) = 256h(x^*)$ (s. [Mis98], S.20). Die Wahrscheinlichkeit, derartige Dokumente zu finden lag in diesem Falle bei 1 zu 4096, d.h. extrem niedrig. Jedoch konnte somit aufgrund der Multiplikativität des RSA-Algorithmus $S(h(x)) = S(256)S(h(x^*))$ folgende Signaturfälschung erfolgen:
Findet ein Angreifer drei Dokumente x_1, x_2, x_3, mit $h(x_1) = 256h(x_2)$ und $h(x) = 256h(x_3)$, und gelingt es ihm zudem sich zu x_1 und x_2 die Originalsignaturen zu beschaffen, so kann er den Wert von $S(256)$ bestimmen ($S(256) = \frac{S(h(x_1))}{S(h(x_2))}$) und somit auch die Signatur von x_3. Damit erhält der Widersacher zugleich die Signatur zu dem gewünschten x: $S(h(x)) = S(256)S(h(x_3)) = \frac{S(h(x_1))}{S(h(x_2))}S(h(x_3))$.

Insgesamt kann man durch entsprechende Sicherheitsvorkehrungen, wie Geheimhaltung und entsprechende bedachte Wahl der Primfaktoren und dem Einbau von Hash- bzw. Redundanzfunktionen davon ausgehen, daß das RSA-Signaturverfahren sicher vor Angriffen ist und eine gute Möglichkeit darstellt, Dokumente zu signieren. Ein weiteres, häufig angewandtes Signaturverfahren ist das ElGamal-System, das im folgenden vorgestellt wird.

7.4 Das ElGamal-Signatursystem

Das *ElGamal-Signaturverfahren* ist ein System, welches die Grundlage für den *DSA* (Digital Signature Algorithm) (s. 7.6.2) darstellt. Seinerseits baut es auf dem *ElGamal-Verschlüsselungsverfahren* auf. Dieses ist ein *probabilistisches Public-Key-Verfahren*, welches auf der *diskreten Logarithmusfunktion* basiert. Seine Sicherheit beruht zum einen darauf, daß der zugehörige Algorithmus außer von dem Klartext und dem verwendeten öffentlichen Schlüssel auch von der Wahl eines Zufallsobjekts abhängt, daher *probabilistisch* ist, sowie auf der Schwierigkeit, diskrete Logarithmen in einem endlichen Körper zu berechnen. Diese Tatsache beschreibt das *Diskrete Logarithmusproblem* in Z_p:
Sei α ein Erzeuger von Z_p^* und auch $\beta \in Z_p^*$. Dann heißt der eindeutig bestimmte Exponent a, $0 \leq a \leq \varphi(p)$ mit $\alpha^a \equiv \beta (mod\ p)$ Index oder *diskreter Logarithmus modulo p* von β zur Basis α (kurz: $a = log_{p,\alpha}(\beta)$). (s. auch 7.2)
Geg.: $p\ prim,\ \alpha, \beta \in Z_p^*$
Ges.: $a,\ 0 \leq a \leq p-2$, so daß gilt: $\alpha^a \equiv \beta\ (mod\ p)$.
Sei $a := \log_\alpha \beta$
Angenommen p ist *prim* und $\alpha \in Z_p$ sei fest. Dann stellt sich das diskrete Logarithmusproblem folgendermaßen dar: Bei gegebenem $\beta \in Z_p^*$ wird der Exponent a gesucht, wobei $0 \leq a \leq p-2$, so daß gilt: $\alpha^a \equiv \beta\ (mod\ p)$.
Die Funktion $\beta \equiv \alpha^a\ mod\ p$ ist nach heutigem Kenntnisstand zur sicheren Verschlüsselung und zum Signieren geeignet, da der *diskrete Logarithmus* a zu einer vorgegebenen Zahl β nur schwer zu ermitteln ist.

Der ElGamal-Algorithmus als Signaturverfahren
Zur *Schlüsselgenerierung:* wählt man beim ElGamal-Verfahren eine Primzahl p und zwei Zufallszahlen α und a, ($\alpha \leq p$ und $a \leq p$). Dann berechnet man die Funktion $\beta \equiv \alpha^a\ mod\ p$, wobei $\alpha \in Z_p^*$ sein sollte. β, α und p sind öffentlich bekannt, a ist geheim.
Die Signaturgenerierung und -verifikation laufen wie folgt ab:
Zum *Signieren* eines Dokuments x mit ElGamal geht Bob wie folgt vor:
a) Er wählt eine Zufallszahl k die *relativ prim* zu $p-1$ ist, d.h. $k \in Z_{p-1}^*$
b) Dann berechnet Bob eine Lösung δ der Kongruenz $a\gamma + k\delta \equiv x\ (mod\ p-1)$
Dies kann zum Beispiel dadurch geschehen, daß er mit Hilfe des erweiterten euklidischen Algorithmus das multiplikative Inverse k^{-1} zu k *modulo* $p-1$ berechnet und daraufhin: $\delta = (x - a\gamma)k^{-1}\ (mod\ p-1)$ bestimmt. Die Unterschrift zu der Nachricht x besteht dann aus dem Paar (γ, δ), d.h. $S(x,k) = (\gamma, \delta)$. Hierbei muß die Zufallszahl k geheimgehalten werden(s.u.).
Zur *Verifikation* der Unterschrift berechnet Alice die Werte :
$\beta^\gamma \gamma^\delta\ mod\ p$ und $\alpha^x (mod\ p)$ und vergleicht, ob diese identisch sind.
D.h. gilt für $x, \gamma \in Z_p^*$ und $\delta \in Z_{p-1}$:
$V(x, \gamma, \delta) = echt$, also $\Leftrightarrow \beta^\gamma \gamma^\delta \equiv \alpha^x (mod\ p)$, so ist die Signatur authentisch, ansonsten wird sie nicht von Alice anerkannt.

Beweis für die Gültigkeit der Verifikationsfunktion:
Tatsache ist, daß $a\gamma + k\delta \equiv x(mod\ (p-1))$ (s. b))
$\alpha^{a\gamma}\alpha^{k\delta}(mod\ p) = \alpha^{(a\gamma+k\delta)}(mod\ p) =$
$\alpha^{x\ mod(p-1)}(mod\ p) = \alpha^{x+i(p-1)} = \alpha^x(\alpha^{p-1})^i$
Laut dem *Satz von Fermat*[4] $\Rightarrow \alpha^x(\alpha^{p-1})^i = \alpha^x(mod\ p), i \in Z$ [Sti95], S.207.

Sicherheit von ElGamal:
Die Sicherheit des ElGamal-Verfahrens hängt von der verwendeten Logarithmusfunktion ab. Angenommen ein Angreifer möchte die Signatur zu einem Dokument x fälschen, ohne a zu kennen. Wählt dieser Angreifer ein γ und versucht das dazu korrespondierende δ zu finden, so muß er den diskreten Logarithmus: $log_\gamma \alpha^x \beta^{-\gamma}$ berechnen. Wählt er zuerst δ und versucht dann, γ zu finden, so muß er die Gleichung $\beta^\gamma\gamma^\delta \equiv \alpha^x(mod\ p)$ für das „unbekannte" γ lösen. Dies ist ein Problem, für das bisher keine effiziente Lösung bekannt ist. [Sti95], S.207.
Es bliebe noch die Möglichkeit, γ und δ gleichzeitig so zu berechnen, daß (γ, δ) eine Signatur bilden. Bisher hat jedoch noch niemand einen Weg gefunden, derartige Berechnungen durchzuführen, aber es konnte auch noch niemand beweisen, daß dies nicht möglich ist.
Ist jedoch die Zufallszahl k bekannt, so ist es einfach mit ihrer Hilfe den privaten Schlüssel $a \equiv (x - k\delta)\gamma^{-1}\ mod\ (p-1)$ zu berechnen, womit das System gebrochen ist. Der Angreifer kann dann Unterschriften nach Belieben fälschen. Desweiteren ist für jede ElGamal-Unterschrift oder -Verschlüsselung eine neue Zufallszahl k zu wählen, da ansonsten anhand zweier Dokumente, die mit demselben k verschlüsselt wurden, der geheime Schlüssel a bestimmt werden kann.
Indem man zwei Signaturen (γ, δ_1) für ein Dokument x_1 und (γ, δ_2) für ein Dokument x_2 kennt, kennt man auch die entsprechenden Formeln:
$\beta^\gamma\gamma^{\delta_1} \equiv \alpha^{x_1}(mod\ p)$
$\beta^\gamma\gamma^{\delta_2} \equiv \alpha^{x_2}(mod\ p)$
Daraus ergibt sich durch entsprechende Umformungen $\alpha^{x_1-x_2} \equiv \gamma^{\delta_2-\delta_1}(mod\ p)$
Da $\gamma = \alpha^k$ erhält man folgende Gleichung mit unbekanntem k:
$\alpha^{x_1-x_2} \equiv \alpha^{k(\delta_2-\delta_1)}(mod\ p)$
Dies ist äquivalent zu $x_1 - x_2 \equiv k(\delta_2 - \delta_1)(mod\ (p-1))$.
Sei nun $d = ggT(\delta_2 - \delta_1, p-1)$.
Da $d \mid p-1$ und $d \mid (\delta_2 - \delta_1)$, folgt daraus, daß $d \mid (x_1 - x_2)$.
Definiere nun $x^* = \frac{x_1-x_2}{d}$, $\delta^* = \frac{\delta_2-\delta_1}{d}$ und $p^* = \frac{p-1}{d}$.
Hiermit wird obige Kongruenzgleichung zu: $x^* \equiv k\delta^*\ (mod\ p^*)$.
Da der $ggT(\delta^*, p^*) = 1$ kann man das multiplikative Inverse zu δ^* folgendermaßen berechnen: $\epsilon = (\delta^*)^{-1}\ mod\ p^*$
Dann gilt für $k\ modulo\ p^*$: $k = x^*\epsilon\ mod\ p^*$
D.h. es gibt d mögliche Werte für k: $k = x^*\epsilon + ip^*\ mod\ p$, für $i: \ 0 \le i \le d-1$.
Der richtige Wert für k kann dann anhand der Bedingung $\gamma \equiv \alpha^k\ (mod\ p)$ gefun-

[4] $a^{p-1} \equiv 1\ (mod\ n)$, für $a \in Z_p^*$, $p\ prim.$ (s. 7.2)

den werden. Erfüllt k diese Gleichung, so hat man den gesuchten k-Wert ermittelt. Aufgrund der Ermittlung von k, kann dann, wie bereits erwähnt a berechnet und somit das System gebrochen werden. [Sti95], S.206.

Beim ElGamal-System ist folglich darauf zu achten, die Zufallszahl k geheim zu halten und zudem für jede Signatur ein neues k zu bestimmen. Je besser also der Zufallsgenerator, desto sicherer das System. Da die Sicherheit dieses Signaturverfahrens auf der Sicherheit der Logarithmusfunktion basiert, sollte zudem eine möglichst große Primzahl für p gewählt werden (mittlerweile werden Primzahlen ab 512 Bit vom NIST empfohlen), da für diese Fälle bisher noch kein effizienter Berechnungsalgorithmus bekannt ist.

7.5 Die MD5-Hashfunktion und der „Secure Hash Algorithm"

Unter einer sicheren Hashfunktion versteht man eine Einweghashfunktion, die aufgrund ihrer Einwegeigenschaft (nach dem heutigen Kenntnisstand) gute Sicherheit gegen Angriffe auf Signaturverfahren, wie in *Kapitel 4* geschildert wurde, gewährleisten kann. Im folgenden wird zunächst die MD5-Hashfunktion als sichere Einweghashfunktion vorgestellt, da darauf der SHA-1 (Secure Hash Algorithm) basiert und häufig praktische Anwendung findet.
Der „MD5-Algorithmus" (Abkürzung für Message Digest5) wurde von Ron Rivest am MIT entwickelt. Der Algorithmus nimmt als Eingabe ein Dokument beliebiger Länge und erzeugt als Ausgabe einen Hash-Code von 128 Bit Länge. Dieser Algorithmus wird häufig einem Signaturverfahren „vorangestellt", um Signaturverfahren anwenden zu können, die nur Eingaben fester Länge verabeiten können und um eine Effizienzsteigerung zu bewirken (s. *Kapitel 4*).
Bei der MD-5-Hashfunktion wird die Eingabe in Blöcken zu 512 Bit und zu 16 Teilblöcken der Länge 32 Bit abgearbeitet. Die Ausgabe des Algorithmus besteht aus vier 32 Bit-Blöcken, die konkateniert den 128 Bit-Hashwert ergeben. Dabei sind folgende Schritte durchzuführen:

1 Anfügen von Füllbits:
Das Dokument wird so aufgefüllt, daß die Länge einem Vielfachen von 512 minus 64 Bit entspricht.

2 Anfügen der Länge:
Anschließend wird eine 64 Bit lange Darstellung der Länge in Bit des ursprünglichen Dokuments (ohne Füllbits) angefügt.

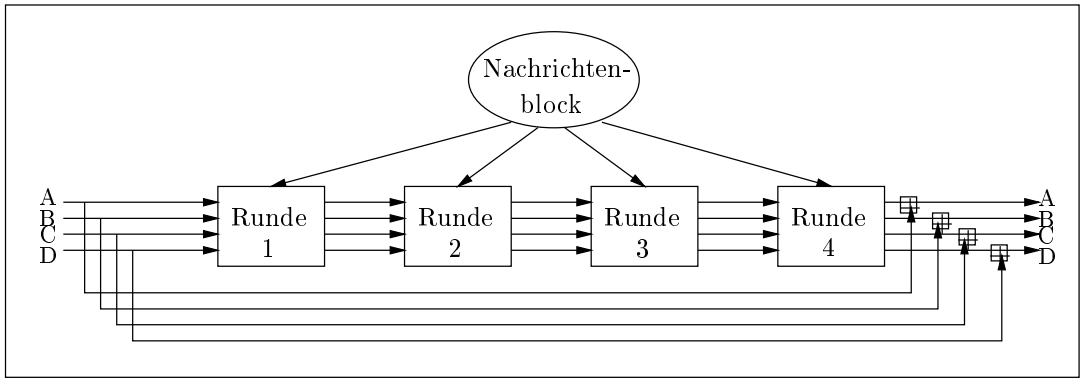


Abbildung 7.2: Darstellung der Hauptschleife des MD5-Algorithmus

3 Initialisierung des MD-Puffers:
Um die Zwischen- und Endergebnisse der Hashfunktion „festzuhalten", wird ein 128 Bit-Puffer verwendet. Dieser kann durch vier 32 Bit-Register dargestellt werden (A,B,C,D). Diese Variablen heißen Verkettungsvariablen.

4 Verarbeitung der Dokument in 512 Bit-(16 Wort-)Blöcken:
In der Hauptschleife des Algorithmus (s. Abb. 7.2) durchlaufen alle 512 Bit-Blöcke die gleichen Verarbeitungsschritte. Dieser besteht aus vier Runden, die sich alle sehr ähnlich sind. Jede Runde benutzt 16 mal jeweils eine andere Operation. Jede Operation führt mit drei der vier Werte A,B,C und D eine nichtlineare Funktion durch. Dieses Ergebnis wird zur vierten Variablen, einem Teilblock des Texts und einer Konstanten addiert. Daraufhin erfolgt eine Rotation einer variablen Anzahl von Bits nach rechts und das Ergebnis wird zu einem der Werte A,B,C oder D addiert. Daraufhin ersetzt das Ergebnis einen der Werte A, B, C oder D.
A, B, C und D werden mit folgenden hexadezimalen Werten vorbelegt:
A=0123456 B=89ABCDEF C=FEDCBA98 D=76543210

5 Ausgabe:
Die Ausgabe bildet letztendlich eine Konkatenation der Werte A, B, C und D.
MD5 wird in vielen Anwendungen verwendet, z.B. bei PGP (s. *Kapitel 6*) und scheint u.a. aufgrund praktischer Erfahrungen äußerst sicher zu sein ([Sta95], S.242).

Der Secure Hash Algorithm *SHA-1* wurde vom U.S. National Institute for Standards and Technology (NIST) für bestimmte U.S. Regierungs-Applikationen vorgeschlagen. Er basiert sehr stark auf dem MD5-Algorithmus und findet in vielen Digitalen Signatur-Schemata Anwendung, v.a. auch im DSS (Digital Signature Standard) und gilt als sehr sicher.

Auch beim SHA-1 wird zunächst das Dokument so erweitert, daß die Länge ein Vielfaches von 512 Bit beträgt. Dieser Vorgang verläuft analog zum MD5. Daraufhin werden fünf Variablen der Länge 32 Bit initialisiert (s. [Sch96], S.505) und in verschiedene Variablen kopiert. In der Hauptschleife werden in vier Runden jeweils 20 Operationen ausgeführt, wobei jede Operation mit drei der fünf Variablen eine nichtlineare Funktion durchführt. Anschließend finden ähnlich wie beim MD5 Verschiebungen und Additionen statt.
Der SHA erzeugt für jede Eingabe der Länge $\leq 2^{64}$ Bit, d.h. für beliebig große Eingaben, eine Ausgabe der Länge 160 Bit. Insgesamt verwendet der SHA-1 eine Variable mehr als der MD5, führt vier Runden mehr in der Hauptschleife durch und erzeugt um 32 Bit längere Hashwerte. Die wichtigsten Erweiterungen des MD5 sind durch das Hinzufügen einer Expansionstransformation und der Einbezug der Ausgabe des vorherigen Schritts in den nächsten Schritt bestimmt. Diese Maßnahmen zielen darauf ab, die Sicherheit, die durch den MD5 bereits gewährleistet wird, noch zu erhöhen.

7.6 Der Digitale Signatur-Algorithmus (*DSA*)

Am 19. Mai 1994 wurde im Federal Register der *DSS* (Digital Signature Standard) als Standard für Digitale Signaturen veröffentlicht und endgültig am 01. 12. 1994 als solcher verabschiedet. Mit dem vorgeschlagenen *DSS* ist es dem Verifizierer eines Dokuments möglich, mit Hilfe des öffentlichen Schlüssels des Unterzeichners die Integrität der gesendeten Daten und die Identität des Unterzeichners zu verifizieren.
Der *DSS* spezifiziert einen Public-Key-Algorithmus für Digitale Signaturen, den *DSA*. Dieser Algorithmus stellt eine Variante des ElGamal-Signaturverfahrens dar und wurde im August 1991 vom NIST[5] für den Einsatz im *DSS* erstmals vorgeschlagen.

7.6.1 Herleitung des *DSA* vom ElGamal-Signaturverfahren

Der ElGamal-Algorithmus ist maximal so sicher, wie das Diskrete Logarithmusproblem, welches einen großen Primfaktor als Modulus zur Sicherheit benötigt. Dieser sollte mindestens 512 Bit aufweisen (s. Sicherheit von *DSA* 7.6.3). Bei ElGamal führt diese Modulo-Größe zu einer doppelt-so-großen, d.h. 1024 Bit-Signatur. Für viele Anwendungen, die Digitale Signaturen verwenden möchten, wie z.B. Signieren mittels SmartCards, wären kürzere Signaturen wünschenswert. Beim *DSS* wird das ElGamal-Signatursystem dementsprechend modifiziert, so daß 160 Bit-Dokumente signiert eine Signatur von 320 Bit ergeben, hierbei jedoch zur Berechnung einen Modulus von 512 Bit verwenden können.

[5] National Institute for Standards & Technology

Dies kann dadurch bewerkstelligt werden, indem man in einer Untergruppe von Z_p^* der Größe 2^{160} arbeitet. Man nimmt hierbei an, daß man vor der Suche diskreter Logarithmen in dieser Untergruppe auch sicher ist. Dazu verändert man das „-" in der Definition von δ zu einem „+":
$\delta = (x + \alpha\gamma)k^{-1} \ (mod\ p-1)$
Dies verändert die Verifikationsbedingung folgendermaßen:
$\alpha^x \beta^\gamma \equiv \gamma^\delta \ (mod\ p)$ (Gleichung 1)
Ist der $ggT(x + \alpha\gamma, p-1) = 1$, dann existiert $\delta^{-1} \ (mod\ p-1)$ (s. 7.2: Multiplikatives Inverses) und man gelangt zu folgender Gleichung:
$\alpha^{x\delta^{-1}} \beta^{\gamma\delta^{-1}} \equiv \gamma \ (mod\ p)$ (Gleichung 2)
Dies ist die größte Neuerung von *DSA* im Vergleich zum ElGamal-Signaturalgorithmus. Angenommen q wäre eine 160 Bit-Primzahl, so daß $q/(p-1)$ und α wäre die q-te Wurzel von (1 *modulo* p).[6] Dann stellen auch β und γ die q-te Wurzel von 1 dar. Daher kann jeder beliebige Exponent von α, β und γ durch *modulo* p reduziert werden, ohne die Gleichung 1 zu verändern. Im folgenden wird nun der DSA vollständig vorgestellt.

7.6.2 Der Digitale Signatur-Algorithmus (*DSA*)

Der DSA verwendet folgende Paramter:
$p =$ eine Primzahl der Länge l Bit, wobei l von 512 bis zu 1024 Bit ein Vielfaches von 64 sein kann, so daß das diskrete Logarithmusproblem in Z_p nicht lösbar ist.
$q =$ ein 160 Bit langer Primfaktor von $p-1$ ist;
Sei $\alpha \in Z_p^*$ die q-te Wurzel von 1 *modulo* p. Sei $X = Z_p^*$, $Y = Z_q \times Z_q$
Sei $\beta \equiv \alpha^a (mod\ p)$, wobei p, q, α und β öffentlich sind und a geheim.
Sei k eine Zufallszahl, mit $1 \leq k \leq p-1$ und definiere die Signaturfunktion durch $S(x,k) = (\gamma, \delta)$, wobei $\gamma = (a^k \ mod\ p) \ mod\ q$ und $\delta = (x + a\gamma)k^{-1} \ mod\ q$.
Für $x \in Z_p^*$ und $\gamma, \delta \in Z_q$, wird die Verifikation einer Signatur durch folgende Berechnung durchgeführt:
$e_1 = x\delta^{-1} \ mod\ q$ und $e_2 = \gamma\delta^{-1} \ mod\ q$
$V(x, \gamma, \delta) = echt \Leftrightarrow (\alpha^{e_1}\beta^{e_2} \ mod\ p) \ mod\ q = \gamma$.
Anm.: Es ist notwendig, daß $\delta \neq 0 \ (mod\ q)$, da der Wert $\delta^{-1} \ mod\ q$ benötigt wird, um eine Signatur verifizieren zu können. Dies entspricht der Anforderung, daß der $ggT(\delta, p-1) = 1$ ist. Berechnet Bob einen Wert für δ mit $\delta \equiv 0 (mod\ q)$ beim Signaturalgorithmus, so sollte er eine neue Signatur mit einer neuen Zufallszahl k berechnen. In der Praxis ist die Wahrscheinlichkeit, daß dies auftritt, 2^{-160}, d.h. vernachlässigbar gering.

[6] Ein derartiges α ist einfach zu konstruieren: Sei α_0 ein Erzeuger von Z_p, so definiere $\alpha = \alpha_0^{(p-1)/q} \ mod\ p$

Zur **Schlüsselgenerierung:** generiert jede „Einheit" einen geheimen und öffentlichen Schlüssel wie folgt:
1. Wähle eine Primzahl q, so daß $2^{159} \leq q \leq 2^{160}$
2. Wähle t, so daß $0 \leq t \leq 8$ und finde eine Primzahl mit $2^{511+64t} \leq p \leq 2^{512+64t}$, mit der Eigenschaft: q teilt $p-1$.
3. Wähle einen Erzeuger α einer eindeutig zyklischen Gruppe der Ordnung q in Z_p^*.
3.1 Wähle ein Element $\alpha \in Z_p^*$ und berechne $\beta = \alpha^a \ mod \ p$
3.2 Falls $\alpha = 1$, so gehe zu Schritt 3.1
4. Wähle eine Zufallsintegerzahl k, so daß $1 \leq k \leq q-1$.
5. Berechne $\beta = \alpha^k \ mod \ p$.
6. Der öffentliche Schlüssel von Alice ist dann: (p, α, β),
der geheime Schlüssel ist a.

Ablauf des Signierens:
Alice signiert eine Dokument x beliebiger Länge wie folgt:
a) Wähle eine Zufallsintegerzahl $k, 0 \leq k \leq q$.
b) Berechne $\gamma = (\alpha^k \ mod \ p) \ mod \ q$
c) Berechne $\delta = k^{-1}(h(m) + x\gamma) \ mod \ q$
d) Die Signatur von Alice für x ist dann das Paar: (γ, δ)

Ablauf der Verifikation:
Bob verifiziert die Unterschrift (γ, δ) zum Dokument x folgendermaßen:
a) Bob erhält den Öffentlichen Schlüssel (p, α, β)
b) Gilt $0 \leq \gamma \leq q$ und $0 \leq \delta \leq q$, so akzeptiert Bob die Signatur; ansonsten gilt, daß die Signatur ungültig ist.
c) Berechne $w = \delta^{-1} \ mod \ q$ und $h(x)$
d) Berechne $u_1 = w * h(m) \ mod \ q$ und $u_2 = \delta w \ mod \ q$
e) Berechne $v = (\alpha_1^u \beta_2^u \ mod \ p) \ mod \ q$
f) Gilt $v = \gamma$, so akzeptiert Bob die Signatur; ansonsten weist er sie zurück.

7.6.3 Sicherheit des *DSA*

Da die Sicherheit des *DSA*, wie auch ElGamal von der Unlösbarkeit des Logarithmusproblems ausgeht, ist diese u.a. von der Länge des Primfaktors p abhängig. Möchte man mittels *DSA* kurzlebige Signaturen generieren, so gilt die Länge von 512 Bit für p derzeit als ausreichend sicher. Längerfristig empfiehlt das NIST mindestens eine Länge von 1024 Bit ([Sti95], S. 211.).
Analog zum ElGamal-Signaturverfahren erfordert jede Signatur, die mit dem *DSA* erstellt wird, zudem einen neuen Wert für die Zufallszahl k. Sobald dieser Wert bekannt ist, kann daraus der private Schlüssel ermittelt werden, wodurch Unterschriften der betreffenden Person gefälscht werden können, ohne den ge-

ringsten Zweifel an der Urheberschaft zu erwecken. Analog zum Angriff gegen das ElGamal-System kann durch die mehrfache Verwendung der gleichen Zufallszahl der geheime Schlüssel ermittelt werden, wodurch es für die *DSA*-Algorithmus-Sicherheit entscheidend ist, daß der Zufallszahlengenerator gut ist.

In den vorherigen Abschnitten wurden Algorithmen vorgestellt, die zur Digitalen Signaturbildung häufig verwendet werden, da sie aufgrund ihrer Sicherheitsvorkehrungen für die Anwendung in der Praxis besonders geeignet erscheinen. Im folgenden wird anhand des Beispiels von sog. „Blinden Signaturen" erläutert, wie diese Algorithmen angewandt werden können, um Digitale Signaturen mit zusätzlicher Funktionalität generieren zu können.

7.7 „Blinde Signatur" im Elektronischen Zahlungsverkehr

Unter die Bezeichnung „Electronic Cash" fällt alles, was mit elektronischem Zahlungsverkehr zu tun hat. Nachdem der Begriff zunächst nur im Internet verwendet wurde, ist er inzwischen in den allgemeinen Sprachschatz übergegangen und wird für nahezu jeden, bargeldlosen Zahlungsverkehr verwendet. [www99b]
Im Zusammenhang mit Digitalen Signaturen ist v.a. der Bereich des „Electronic Cash" interessant, der sich mit dem Generieren und Übermitteln virtueller, digital signierter Münzen beschäftigt. Ein Kunde möchte sich von seiner Bank virtuelles Geld auszahlen lassen, welches er dann „anonym" ausgeben kann. Wird das elektronische Geld der Bank als Zahlungsmittel vorgelegt, so sollte die Bank nicht mehr feststellen können, an wen sie das elektronische Geld ausgegeben hatte, aber sie sollte wissen, daß es gültig ist. Diese Vorgehensweise würde es Kunden ermöglichen, anonym zu bleiben, so daß das *Wo* und *Wann* des Geldausgebens nicht protokolliert und ausgewertet werden kann, wie es z.B. für Kaufanalysen in der Wirtschaft oft angestrebt wird.
Wie kann man nun derartige Münzen realisieren? Dies kann man durch die Verwendung „Blinder Signaturen" (engl: Blind Signature Scheme) bewerkstelligen.
Prinzipiell benötigt ein „Blindes Signatur-Protokoll" folgende Komponenten:
Ein Digitales Signaturschema für den Unterzeichner B (z.B. die Bank).
$S_B(x)$ stellt hierbei die Signatur von B zu x dar.
Funktionen f und g, die nur der „Kunde" A kennt, so daß gilt:
$g(S_B(f(x))) = S_B(x)$.
f wird hierbei „blinding function" (verblendende Funktion), g „unblinding function" (entblendende Funktion) und $f(x)$ stellt die „blinded message" (die mit f verblendete Nachricht) dar.

Beispiel für eine „blinding function", die auf RSA basiert:
Sei $n = pq$, wobei p und q große Primzahlen darstellen. Der Signaturalgorithmus S_B sei der RSA-Algorithmus, wie gehabt mit (e, n) als öffentliches Schlüsselpaar und (d, n) als geheimes. Sei nun k eine beliebige, aber feste Integerzahl, mit $ggT(k, n) = 1$.
Die „blinding function" sei definiert als: $f : Z_n \to Z_n$, mit $f(x) = xk^e \ (mod\ n)$
Die „unblinding function" sei $g : Z_n \to Z_n$, mit $g(x) = k^{-1}x \ (mod\ n)$
Für diese Wahl von f,g und S_B gilt dann wie gefordert:
$g(S_B(f(x))) = g(S_B(xk^e \ (mod\ n)) = g(x^d k \ (mod\ n)) = x^d \ (mod\ n) = S_B(x)$
Somit kann A von der Bank eine signierte Münze erhalten, ohne daß die Bank später A diese Münze zuordnen kann. Wie man im einzelnen beim Geldabheben und Einzahlen vorgehen kann, um mehrfaches Ausgeben einer elektronischen Münze vermeiden zu können, wird an folgendem Beispiel erläutert.

Beispiel eines digitalen Bargeldsystems anhand des Nyberg-Rueppel Signaturverfahrens
Auf dem „First International Workshop" zu Informationstheoretischer Sicherheit („Information Security") wurde ein neues „anonymes" digitales Bargeldsystem vorgestellt, welches auf der „Blinden Nyberg-Rueppel" Digitalen Signatur basiert [KQN98]. Das Neue an diesem Verfahren stellt die Tatsache dar, daß ein Kunde Münzen anonym ausgeben kann, jedoch bei mehrfachem Ausgeben ein und derselben Münze durch die Bank ausfindig gemacht werden kann.
Wie ist nun das „Nyberg-Rueppel" Signaturverfahren gestaltet, so daß daraus ein anonymes, elektronisches Bargeldsystem entwickelt werden kann?

Das Nyberg-Rueppel Digitale Signaturverfahren
Sei x der private Schlüssel des Unterzeichners (der Bank) und $y = g^x \ mod\ p$ der zugehörige öffentliche Schlüssel, wobei $g \in Z_p$ und $p\ prim.$ Zudem benötigt man eine zusätzliche Primzahl q, so daß $g^q = 1 \ mod\ p$ gilt. g, p, q sind dabei öffentlich bekannt.
Um nun eine Nachricht $m \in Z_p$ zu unterschreiben, wählt der Unterzeichner eine Zufallszahl $k \in Z_p$ und berechnet r und s folgendermaßen:
$r = mg^k \ mod\ p$ und $s = xr + k \ mod\ q$. Das Paar (r, s) bildet die Signatur zu der Nachricht m.
Um die Signatur zu verifizieren, testet man die Gültigkeit von $m = g^{-s}y^r r \ mod\ p$.
Beweis für die Gültigkeit dieser Verifikation:
$g^{-s}y^r r \ mod\ p = g^{-xr-k(mod\ q)}g^{xr}mg^k \ (mod\ p) =$
$mg^{-xr-k+iq}g^{xr}g^k \ (mod\ p) = mg^{-xr-k+iq+xr+k} \ (mod\ p) = (i \in Z)$
$mg^{iq} \ (mod\ p) = m$, da laut Vor.: $g^q = 1 \ mod\ p$.

Umwandlung des Nyberg-Rueppel Signaturverfahrens in ein Blindes Signaturschema

Um das Nyberg-RueppelSignaturverfahren in ein Blindes Signaturverfahren umzuwandeln, erhält der Verifizierer ein derartiges Signaturpaar (r, s) vom Unterzeichner, allerdings auf die Art und Weise, daß der Unterzeichner selbst nichts Wesentliches über r und s weiß. Dies kann man durch folgenden Prozeß erreichen:
Der Unterzeichner wählt sich eine Zufallszahl $\overline{k} \in Z_q$, berechnet $\overline{r} = g^{\overline{k}} \ (mod\ p)$ und übermittelt $\overline{r}$ an den Verifizierer.
Der Verifizierer wählt sich nun zwei Zufallszahlen $\alpha, \beta \in Z_q$, berechnet
$r = mg^{\alpha}\overline{r}^{\beta} \ (mod\ p)$, $\overline{m} = r\beta^{-1} \ mod\ q$ und sendet $\overline{m}$ zum Signierer.
Der Unterzeichner berechnet nun $\overline{s} = \overline{m}x + \overline{k} \ mod\ q$ und übermittelt $\overline{s}$ an den Verifizierer.
Dieser bestimmt sodann $s = \overline{s}\beta + \alpha \ (mod\ q)$.
Das Paar (r, s) stellt somit eine „blinde Signatur" des Unterzeichners zu einer Nachricht m dar. Die Verifikation der Signatur erfolgt mittels der Berechnung:
$g^{-s}y^{r}r = mg^{-\overline{s}\beta+xr+\overline{k}\beta+\alpha} = mg^{-\overline{m}x\beta-\overline{k}\beta+xr+\overline{k}\beta} =$
$mg^{-r\beta^{-1}x\beta-\overline{k}\beta+xr+\overline{k}\beta} = m \ (mod\ p)$.
Da α und β zufällig gewählt werden, kann der Unterzeichner nichts von r und s wissen. Zu einer gegebenen Signatur (r, s), existiert nur ein eindeutiges Paar (α, β). (lt. [KQN98], S.314). Dies bedeutet, daß der Verifizierer zu jeder Signatur des Unterzeichners nur eine blinde Signatur generieren kann. (s. auch: Sicherheit).
Im folgenden wird nun erläutert, wie man aus diesem Blinden Signaturverfahren, ein Digitales Zahlungssystem entwickeln kann.

Umsetzung des Nyberg-Rueppel Signaturverfahrens in ein Digitales Zahlungssystem

Seien nun B die Bank, C der Klient und V der Händler. Wie bisher gilt: (p, q, g) sind öffentlich bekannt und erfüllen die Bedingung: $g^q = 1 \ mod\ p$.
B arbeitet mit dem geheimen Schlüssel x und wählt zwei Zufallszahlen w_1, w_2 und berechnet $g_1 = g^{w_1} \ mod\ p$, $g_2 = g^{w_2} \ mod\ p$, sowie $h_1 = g_1^x \ mod\ p$ und $h_2 = g_2^x \ mod\ p$. g_1, g_2, w_1, w_2 werden öffentlich bekannt gemacht.
Jeder Klient verfügt über ein Schlüsselpaar aus einem geheimen und öffentlichen Schlüssel. Dies seien u und v, mit $v = g_1 g_2^u \ mod\ p$. Der öffentliche Schlüssel v wird bei der Bank als die Identität von C registriert.
Die Bank ihrerseits stellt C ein Zertifikat für dessen „Identität" aus: $w = v^x \ mod\ p$.
Das System wird nun in drei Phasen unterteilt:
(Geld-)Abhebung, Zahlung und Bankeinzahlung. C hebt zunächst elektronische Münzen von seiner Bank ab, gibt diese bei einem Händler aus, welcher sie anschließend bei der Bank wieder einzahlt und seinem Konto gutschreiben läßt. Diese Vorgänge werden mittels des sog. Abhebungs-, Zahlungs- und Bankeinzahlungsprotokoll wie folgt beschrieben.

1 Abhebungs-Protokoll
B wählt eine Zufallszahl $k \in Z_q$, berechnet $\delta = v^k \ mod \ p$ und leitet δ an C weiter.
C generiert drei Zufallszahlen (y, x_1, x_2), berechnet $\alpha = w^y \ mod \ p$,
$\beta = v^y \ mod \ p$ und $\lambda = h_1^{x_1} h_2^{x_2} \ mod \ p$.
C bildet die Nachricht $m = H(\alpha, \beta, \lambda)$, generiert eine Zufallszahl a und einen Nyberg-Rueppel- Verblendungsfaktor b, berechnet $r = m\beta^a \delta^{bw} \ mod \ p$ und sendet $\overline{m} = r/b \ mod \ q$ zu B.
H stellt hierbei eine stark-kollisionsresistente Einweghashfunktion dar.
B berechnet seine Nyberg-Rueppel Signatur zu der „verblendeten" Nachricht $\overline{m}$ durch Berechnung von $\overline{s} = \overline{m}x + k \ mod \ q$ und sendet diese an C.
C entfernt den Verblendungsfaktor b und erhält die Signatur
$s = \overline{s}b + a = (\overline{m}x + k)b = rx + kb + a(\ mod \ q)$
Am Ende dieses Protokolls bildet $[\alpha, \beta, \lambda, r, s]$ eine gültige Münze. Sie kann mittels $H(\alpha, \beta, \lambda) = \beta^{-s}\alpha^r r \ mod \ p$ verifiziert werden.

Annahme 1 (Vollständigkeit):
Führen C und B die Berechnungen korrekt durch, so gilt die Verifikationsgleichung.
Beweis für die Gültigkeit der Verifikationsgleichung:
Geg.: $s = \overline{s}b + a$, $\overline{s} = \overline{m}x + k$ und $\overline{m} = r/b$. Daher ist $s = ((r/b)x + k)b + a = rx + kb + a \ mod \ p$.
Zudem ist $r = m\beta^a \delta^{yb} \ mod \ p$ äquivalent zu mv^{ay+kby}.
Daher gilt: $\beta^{-s}\alpha^r r = (v^y)^{-s}(w^y)^r r = (v^y)^{-rx-kb-a}(v^{xy})^r r =$
$v^{(-rxy-kby-ay)} v^{rxy} r = v^{(-kby-ay)} m v^{ay+kby} = m = H(\alpha, \beta, \lambda) \ (mod \ p)$

Annahme 2:
Angenommen die Sicherheit der Nyberg-Rueppel Signatur besteht, so kann ein Klient nicht mehr als eine gültige Münze generieren, die den Verifikationsalgorithmus besteht.
Beweis:
Angenommen es wäre möglich, mehr als eine gültige Münze pro „Abhebung" zu generieren. Da H eine starke Einweghashfunktion darstellt, müßte der Klient eine Methode finden, zu der Nyberg-Rueppel Signatur zu einer Nachricht m: $m = H(\alpha, \beta, \gamma)$, die er von einem Mal Geldabheben erhalten hat, eine gültige Signatur zu einer anderen Nachricht $\overline{m} = H(\overline{\alpha}, \overline{\beta}, \overline{\gamma})$ zu generieren.
Dies ist jedoch nicht möglich, wenn man davon ausgeht, daß die Nyberg-Rueppel Signatur sicher ist und die Hashfunktion H eine stark kollisionsresistente Einweghashfunktion darstellt.

Annahme 3 (Blindheit):
Nach der Abhebungs-Transaktion, hat die Bank keine spezifischen Informationen über die einzelnen Münzen. Die Münze, die ausgezahlt wurde, ist anonym.
Beweis:
Es ist einleuchtend, daß sich bei der Berechnung der einzelnen Terme der Münze $[\alpha, \beta, \gamma, r, s]$ zumindest eine Zufallszahl befindet, die geheim von dem Klienten gewählt wurde. Dies bedeutet, daß nach dem Abheben, die Bank (und die Händler), keine andere Information über die Münze besitzen, als die Tatsache, daß sie gültig ist und somit die Anonymität gewahrt ist.

2 Zahlungsprotokoll
Möchte C nun die Münze $[\alpha, \beta, \gamma, r, s]$ an den Händler V zahlen, so folgt er folgendem Protokoll:
1. V generiert eine Zufallszahl c und sendet sie zu C, wobei
$c \leftarrow H(V, Datum, Zeit, ...)$. Dieses c sollte für jede Transaktion neu bestimmt werden.
2. Nach erhaltenem c, berechnet C eine Antwort (r_1, r_2), wobei
$r_1 = x_1 + cy \ mod \ q$ und $r_2 = x_2 + ucy \ mod \ q$ und sendet es an V.
3. V kontrolliert die Nypberg-Rueppel Signatur $H(\alpha, \beta, \gamma)$ und verifiziert, ob (r_1, r_2) mit c konsistent ist. Gilt dies, so ist die Münze gültig.
V akzeptiert die Münze, falls $H(\alpha, \beta, \gamma) = \beta^{-s}\alpha^r r = h_1^{r_1} h_2^{r_2} = \alpha^c \lambda$.
V führt dann den angeforderten Service von C durch und speichert
$(c, r_1, r_2, [\alpha, \beta, \gamma, r, s])$.

Annahme 4 (Vollständigkeit):
Falls der Klient C und der Händler V dem beschriebenen Protokoll korrekt folgen, dann müssen $H(\alpha, \beta, \lambda) = \beta^{-s}\alpha^r r$ und $h_1^{r_1} h_2^{r_2} = \alpha^c \lambda$ gelten.
Beweis:
Sendet der Klient C die gültige Münze $[\alpha, \beta, \gamma, r, s]$ an V, so gilt aus Annahme 1 und 2: $H(\alpha, \beta, \lambda) = \beta^{-s}\alpha^r r$
Zu berücksichtigen ist nun noch $\alpha^c \lambda$:
$\alpha^c \lambda = w^{yc} h_1^{x_1} h_2^{x_2} = v^{xyc} h_1^{x_1} h_2^{x_2} = (g_1 g_2^u)^{xyc} h_1^{x_1} h_2^{x_2} = (h_1 h_2^u)^{yc} h_1^{x_1} h_2^{x_2} = h_1^{x_1+cy} h_2^{x_2+cyu} = h_1^{r_1} h_2^{r_2} \ (mod \ p)$ q.e.d.

Annahme 5:
V akzeptiert die Münze, falls $H(\alpha, \beta, \lambda) = \beta^{-s}\alpha^r r$ wahr ist.
Beweis:
Aus Annahme 1 und 2 folgt: $H(\alpha, \beta, \lambda) = \beta^{-s}\alpha^r r$ gilt dann, wenn die Münze $[\alpha, \beta, \gamma, r, s]$ korrekt abgehoben wurde bzw. wenn sie gültig ist.
Gilt $H(\alpha, \beta, \lambda) = \beta^{-s}\alpha^r r$, so muß V die Münze akzeptieren.

Annahme 6:
Angenommen die Münze ist für einen Zufallswert r gültig, so muß der Klient $r_1 = x_1 + cy$ und $r_2 = x_2 + ucy$ zurücksenden, ansonsten gilt $h_1^{r_1} h_2^{r_2} = \alpha^c \lambda$ nicht.
Beweis:
Jede Münze, die von C ausgegeben wird, besitzt eine Nyberg-Rueppel Signatur auf der Basis, daß v der Öffentliche Schlüssel des Klienten darstellt. Da die Münze gültig ist, gilt: $m = \beta^{-s} \alpha^r r$. Daher müssen α und β ein Vielfaches der Basis v^x und v sein. Dies bedeutet $\alpha = v^{xy}$ für eine Zufallszahl y. Daher gilt: $\alpha^c \lambda = v^{xcy} \lambda = (g_1 g_2^u)^{xcy} = h_1^{cy} h_2^{cuy}$
Andererseits gilt, daß wenn die Münze akzeptiert wird, so muß $h_1^{r_1} h_2^{r_2} = \alpha^c \lambda$ wahr sein. Dies bedeutet: $h_1^{r_1} h_2^{r_2} = h_1^{cy} h_2^{cuy} \lambda$ oder $\lambda = h_1^{r_1 - cy} h_2^{r_2 - cuy} = \alpha^c \lambda$.
Sei $\overline{x}_1 = r_1 - cy$ und $\overline{x}_2 = r_2 - cuy$. Dann gilt: $\lambda = h_1^{\overline{x}_1} h_2^{\overline{x}_2}$.
Das Paar $(\overline{x}_1, \overline{x}_2)$ muß der Klient bereits vor der Transaktion kennen. Zudem ist dieses Paar einzigartig aus der Sichtweise des Klienten, da er/sie $log_{h_1}(h_2)$ nicht kennt.
Analog zur Phase des Geldabhebens, generiert C $\lambda = h_1^{x_1} h_2^{x_2}$. Daher gilt: $x_1 \equiv \overline{x}_1 \ (mod\ p)$; $x_2 \equiv \overline{x}_2 \ (mod\ p)$ und
$r_1 = x_1 + cy \ (mod\ p)$; $r_2 = x_2 + cuy \ (mod\ p)$.

3 *Einzahlungs-Protokoll:*
V kann die Münzen, die er/sie erhalten hat, jederzeit bei der Bank einzahlen.
Für jede erhaltene Münze sendet V $(c, r_1, r_2, [\alpha, \beta, \gamma, r, s])$ zu der Bank B.
B kontrolliert ebenso $H(\alpha, \beta, \lambda) = \beta^{-s} \alpha^r r$ und $h_1^{r_1} h_2^{r_2} = \alpha^c \lambda$.
Sind beide Gleichungen erfüllt, so akzeptiert B die Münze.

Wie sieht es nun mit folgenden Sicherheitsaspekten aus?

- *Klienten-Anonymität:*
  Klientenanonymität wird in diesem Schema stets erfüllt:
  Jede Münze wird „blind" von der Bank unterschrieben. Wird die Münze des Klienten an einen Händler gezahlt, so ist es weder für den Händler noch für die Bank möglich, den Klienten mit der Münze in Verbindung zu bringen. Während dem Zahlungsprotokoll muß der Klient unabhängig von der Münze die Antworten $r_1 = x_1 + cy$ und $r_2 = x_2 + cuy$ dem Händler „zeigen". Werden x_1 und x_2 von dem Klienten geheim gewählt, so ist es unmöglich u, v oder w aus der Antwort zu generieren. Dies bedeutet aber, daß die geforderte Anonymität des Klienten stets gewahrt ist.

- *Nichtnachvollziehbarkeit der Transaktionen*
  Dieses Schema behandelt jede Münze separat, d.h. es besteht keine Verbindung zwischen zwei Münzen. Werden zwei Münzen von einem Benutzer abgehoben und bei zwei unterschiedlichen Händlern ausgegeben, so ist es unmöglich irgendeine Art von Verbindung zwischen diesen Transaktionen herzustellen, d.h. die Nichtnachvollziehbarkeit von Transaktionen ist gewährleistet.

- *Aufdeckung Doppelter Verwendung von Münzen:*
  Das Aufdecken doppelter Verwendung von Münzen ist ein wesentliches Ziel jedes Digitalen Signatursystems, obwohl ansonsten die Transaktionen nicht nachvollziehbar sein sollten. Falls C eine Münze $[\alpha, \beta, \gamma, r, s]$ mehrfach ausgegeben hat, so erhält die Bank zwei unterschiedliche Antworten (r_1, r_2) und $(\overline{r}_1, \overline{r}_2)$ für zwei Zufallswerte c, $\overline{c}$, wobei $r_1 = x_1 + cy$, $r_2 = x_2 + cuy$, $\overline{r}_1 = x_1 + \overline{c}y, \quad \overline{r}_2 = x_2 + \overline{c}uy$.
  B kann dann folgendes berechnen:
  $\frac{r_1 - \overline{r}_1}{c - \overline{c}} = \frac{x_1 + cy - x_1 - \overline{c}y}{c - \overline{c}} = y, \quad \frac{r_2 - \overline{r}_2}{c - \overline{c}} = \frac{x_2 + cyu - x_2 - \overline{c}yu}{c - \overline{c}} = yu$
  Aus y und yu kann die Bank leicht u berechnen. Danach bestimmt B $v = g_1 g_2^u$ und findet leicht die Verbindung von diesem Wert zu dem Klienten in der Kundenliste der Bank. Aufgrund dieser Verbindung kann die Bank B den Klienten C aufgrund des „mehrfachen Ausgebens einer Münze" anklagen. Der Beweis ist unabstreitbar, da die geheime Sicherheitsinformation des Klienten nicht aufzudecken ist, es sei denn der Klient hatte eine Münze zuvor doppelt ausgegeben.

- *Münzfälschung:*
  Um eine Münze zu fälschen, muß der Klient eine Nyberg-Rueppel Signatur zu $m = H(\alpha, \beta, \lambda)$ generieren, was laut der Annahme 2 nicht möglich ist. Das Kombinieren mehrerer alter Münzen zu einer neuen ist ebenso unmöglich, da eine Münze aus $m = H(\alpha, \beta, \lambda)$ besteht und H eine strenge Einweghashfunktion darstellt. Daher kann man Münzen nicht fälschen.

An diesem Beispiel sollte aufgezeigt werden, wie Digitale Signaturen in alltäglichen Szenarien, wie z.B. im Electronic Cash-Bereich Anwendung finden können. Im folgenden Kapitel wird dazu noch die rechtliche Absicherung Digitaler Signaturen erörtert, da sie die Grundlage für derartigen Einsatz Digitaler Signaturen darstellen.

Kapitel 8

Digitale Signaturen und ihre Umsetzung in der Praxis

Die bisherigen Kapitel sollten den Leser in den Aufbau, die Funktionsweise und Realisierung Digitaler Signaturen mittels bestimmter Algorithmen einführen. Wie steht es nun um den praktischen Nutzen von Digitalen Signaturen. Dieser konnte zum Teil schon an den konkreten Beispielen wie PGP und PEM (s. *Kapitel 6*) ersichtlich werden. In diesem Kapitel wird dem Leser zusätzlich die rechtliche Grundlage bei der Verwendung Digitaler Signaturen erläutert und zudem die Vielfalt der Nutzungsmöglichkeiten an zwei weitverbreiteten Software-Systemen veranschaulicht.

8.1 Praktische Vorgehensweise beim Erstellen einer Digitalen Signatur

Wie bereits ersichtlich wurde, kann man eine Digitale Signatur als eine Art Siegel für digitale Daten ansehen. In der Praxis finden meist Digitale Signaturschemata Verwendung, die eine Zusammensetzung aus asymmetrischen Verschlüsselungsverfahren und dem Einsatz von Hashfunktionen,wie unter *Kapitel 7* beschrieben, darstellen. Jeder, der seine Daten digital signieren möchte, benötigt hierzu, wie gehabt, ein einmaliges Schlüsselpaar aus privatem und öffentlichen Schlüssel. Diese werden durch Zertifizierungsstellen den natürlichen Personen fest zugeordnet und durch ein *Zertifikat* beglaubigt.
Ein derartiges Zertifikat stellt ebenfalls ein signiertes, digitales Dokument dar, das den jeweiligen öffentlichen Schlüssel sowie den Namen der Person, der er zugeordnet ist (oder ein Pseudonym) enthält, um Impersonationsangriffe (s. *Kapitel 5*) zu verhindern. Das Zertifikat erhält der Signaturschlüssel-Inhaber, so daß er es signierten Daten für deren Überprüfung beifügen kann. „Kommunikationspartner" können die Gültigkeit dieses Zertifikats testen und die Echtheit eines öffentlichen Schlüssels nachweisen.

Zum Unterschreiben verwendet ein Unterzeichner eine Chipkarte, die sogenannte Smartcard[1], die ebenso wie der zertifizierte öffentliche Schlüssel von einer Zertifizierungsstelle ausgehändigt wird. Auf dieser Smartcard sind Informationen über den privaten Schlüssel und die Signiertechnik gespeichert, die der Kartenbesitzer erst in Verbindung mit einer PIN (Personen-Identifikations-Nummer) einsetzen kann, damit niemand außer der dazu berechtigten Person, diese nutzen kann. Die Karte wird z.B. über einen PC mit Chipkartenlesegerät zur Anwendung gebracht, wodurch ein ausgewähltes Dokument mit entsprechender Software signiert werden kann. Der Verifizierer kann dann anhand des mit dem Dokument übermittelten, zertifizierten öffentlichen Schlüssels verifizieren, ob die Signatur des Dokuments authentisch ist. [Ker98]
Im folgenden wird nun auch ein Einblick in die rechtliche Grundlage Digitaler Signaturen gewährt, welche durch die Verabschiedung des Signaturgesetzes im August 1997 erstmals geschaffen wurde.

8.2 Digitale Signaturen und ihre rechtliche Grundlage

Die rasche Entwicklung in der Informations- und Kommunikationstechnik hat neue Möglichkeiten des Informationsaustausches und der wirtschaftlichen Betätigung eröffnet. Viele rechtlich relevante Vorgänge, wie z.B. Warenbestellungen, Zahlungsanweisungen an Banken etc., die bisher „über Papier" abgewickelt wurden, erfolgen bereits zu einem großen Teil auf elektronischem Wege.
Ein Problem, das bei dieser Art von Kommunikation jedoch auftreten kann, ist die Verfälschung der übertragenen oder gespeicherten Daten, ohne daß dies Spuren hinterläßt, geschweige denn nachgewiesen werden kann. Daraus ergab sich der dringende Bedarf nach einer digitalen Lösung, die dem Anspruch nach Sicherheit bei der Dokumentenerstellung, Kommunikation und Archivierung digitaler Daten gerecht wird, d.h. die bei einer offenen Kommunikation (die Teilnehmer müssen sich hierbei nicht kennen) zuverlässig Rückschlüsse auf den Urheber eines Dokuments schließen läßt und die Daten vor unbemerkter Veränderung schützt. Dieser Forderung wurde durch die Verabschiedung des Digitalen Signaturgesetzes durch die Definition einer „gesetzlichen Digitalen Signatur" nachgekommen, denn nur die nachweisliche Sicherheit Digitaler Signaturen erlaubt es bei Rechtsvorschriften, neben der papiergebundenen Schriftform auch die digitale Form zuzulassen ([GM97], S.412f.).

[1] Eine *Smartcard* ist eine „intelligente" Chipkarte, die neben Speicherbausteinen auch einen Prozessor für Berechnungen von Algorithmen besitzt

8.2.1 Ziel des Signaturgesetzes

Ziel des Gesetzes ist es, daß Signaturen als sicher vor Fälschung und signierte Daten als sicher vor Verfälschung gelten können. Hiermit soll ein administrativer Rahmen vorgegeben werden, der es ermöglichen soll, Digitale Signaturen eindeutig bestimmten Personen zuordnen zu können. In dem Gesetz wird von einer bundesweiten Infrastruktur für die Zuordnung der Signaturschlüssel zu natürlichen Personen ausgegangen und der Einsatz geeigneter technischer Komponenten und die Unterrichtung der Signaturschlüssel-Inhaber über die in ihrem eigenen Interesse zu treffenden Maßnahmen vorausgesetzt. Ein Ziel ist hierbei, daß der Aufbau und Betrieb der Infrastruktur privatwirtschaftlich im freien Wettbewerb, jedoch unter behördlicher Kontrolle durch das BSI (Bundesamt für Sicherheit in der Informationstechnik) erfolgen soll. Durch die gesetzliche Regelung soll eine hohe Gesamtsicherheit gewährleistet werden. Dies umfaßt sowohl die Erzeugung der Signaturschlüssel als auch die Zuordnung derselben durch zuverlässige Zertifizierungsstellen bis hin zur Darstellung der zu signierenden Daten. Soweit durch Rechtsvorschrift die Schriftform vorgegeben ist, muß geprüft werden, ob und in welchen Fällen es zweckmäßg erscheint, neben der Schriftform auch digitale Dokumente mit Digitaler Signatur zuzulassen.

8.2.2 Einblicke in das Signaturgesetz

Das Signaturgesetz trat im Zusammenhang mit dem Informations- und Kommunikationsdienste-Gesetz (IuKDG) am 1. August 1997 (BGB1.I, S.1872f.) in Kraft. Das IuKDG stellt ein Rahmengesetz dar, welches als solches das Teledienstegesetz, das Teledienstedatenschutzgesetz, das Signaturgesetz und Änderungen bestehender Gesetze, die den neuen technischen Möglichkeiten angepaßt wurden, umfaßt.
Das Signaturgesetz liefert den Rahmen für die Nutzung Digitaler Signaturen und Zertifikate und findet seine Umsetzung durch die sog. Signaturverordnung. Diese trat im Oktober 1997 aufgrund des Paragraph 16 des Signaturgesetzes in Kraft und beschäftigt sich in 19 Paragraphen u.a. mit folgenden Punkten:

- Regelung der Verfahren zur Erteilung und Rücknahme von Genehmigungen und der Kosten und Antragsverfahren bei der Vergabe von Zertifikaten
- Zuordnung einer elektronischen Identität zu einer Person, d.h. Erzeugung und Speicherung von Signaturschlüsseln und Identifikationsdaten
- Zertifizierungsstelle zum Erzeugen und Laden der Signaturschlüssel in Chipkarten
- Verfahren zur Sperrung von Zertifikaten
- Schutz der technischen Komponenten

- Software zum Signieren und zum Verifizieren von Daten
- Überprüfung neuer Digitaler Signaturen.

Im folgenden wird nicht auf jeden Paragraph explizit eingegangen, sondern nur auf diejenigen, die im bezug auf die praktische Anwendung, als besonders wichtig gelten. Anmerkungen zu den Paragraphen befinden sich in [GM97], S.435 und [Pal98], S.30.

1 Infrastruktur für Digitale Signaturen
Laut dem Signaturgesetz sollen Anwendungen Digitaler Signaturen nicht nur auf geschlossene Benutzergruppen beschränkt bleiben, sondern benutzergruppenübergreifend verwendet werden können. Dazu ist eine gewisse Infrastruktur erforderlich, die von Zertifizierungsstellen bereitgestellt wird. Innerhalb dieser Infrastruktur kann dadurch jeder mit jedem kommunizieren, ohne daß zuvor eine Kommunikationsbeziehung bestanden haben muß. Die Trust Center werden hierarchisch, wie unter *Kapitel 6* geschildert, angeordnet, wobei den übergeordneten lediglich die Aufgabe zukommt, weitere Trust Center zu zertifizieren, wobei einheitliche Verfahren und ein einheitliches Sicherheitsniveau entsprechend dem Signaturgesetz vorgegeben sind. Eine Signatur wird demzufolge nur anerkannt, wenn das Zertifikat von einer lizensierten Zertifizierungsstelle ausgestellt ist. Die jeweils zuständige Behörde erteilt die Lizenz und stellt Zertifikate für die öffentlichen Signaturschlüssel aus. Dabei sollte ein Trust Center als vertrauenswürdiger Dritter (sog. Trusted Third Party) Vertrauen vermitteln, da es die Funktion eines gesetzlich anerkannten, elektronischen Notars in Form von Schlüsselverteilungs-, Archivierungs- und Freischaltedienste übernimmt.
Die Voraussetzung für die Betriebsgenehmigung eines Trust Center sind, laut dem Gesetz, ein fachkundiges Personal, Zuverlässigkeit bei der Abwicklung seiner Aufgaben, ein fundiertes Sicherheitskonzept und regelmäßige Auditierungen. [Ker98]

2 Zertifikate
Das *Signaturschlüssel-Zertifikat* muß laut Signaturgesetz folgende Angaben enthalten: Den Namen des Signaturschlüssel-Inhabers, den zugeordneten öffentlichen Signaturschlüssel, die Bezeichnung der Algorithmen, mit denen der öffentliche Schlüssel des Signaturschlüssel-Inhabers sowie der öffentliche Schlüssel der Zertifizierungsstelle benutzt werden kann, die laufende Nummer des Zertifikats, Beginn und Ende der Gültigkeit des Zertifikats, den Namen der Zertifizierungsstelle und Angaben, ob die Nutzung des Signaturschlüssels auf bestimmte Anwendungen nach Art und Umfang beschränkt ist. [Ker98]

3 Gesetzeskonformität von Signaturen
Signaturen sind dann *gesetzeskonform*, wenn sie im Rahmen vorgegebener Infrastrukturen erzeugt, gespeichert angewendet und überprüft werden können. Sie dienen den Sicherheitszielen wie dem Erkennen der Änderung von Daten und dem Nachweis der Urheberschaft von Daten und Aktionen, aber sie unterstützen nicht die Vertraulichkeit von Daten, d.h. das *Ob* und *Wie* Daten verschlüsselt werden müssen. Gerade dieser Punkt wäre in der Praxis wichtig, um Kompatibilitäten unterschiedlicher Signierdienste gewährleisten zu können. Dies ist allerdings nicht Ziel des Gesetzes. [GM97]

8.3 Praktischer Einsatz in Sofware-Systemen am Beispiel von HBCI und SET

8.3.1 HBCI

Das HBCI ist ein 400-seitiges Regelwerk, welches vom ZKA (Zentralen Kreditausschuß) im November 1996 herausgegeben wurde, um auch von heimischen PCs aus Banktransaktionen veranlassen zu können. Dieser neue Homebanking-Standard soll das „veraltete", aber noch dominierende T-Online (Bildschirmtext/Btx-) Verfahren ablösen. Der ZKA-Standard definiert die Schnittstelle durch sogenannte Geschäftsvorfälle zwischen einem HBCI-konformen Homebanking-Client beim Kunden und dem HBCI-Server der Bank. Anfangs konnten mittels des HBCI nur Saldenabfragen, Kontoauszüge, Einfach- und Sammelüberweisungen durchgeführt werden. Mittlerweile ist es auch möglich, Daueraufträge, Terminüberweisungen, Festgeldanlagen bis hin zu Auslandsüberweisungen per HBCI zu veranlassen. Seit dem 2.3.99 ist mit der Verabschiedung der Version 2.1 auch die Wahrnehmung von Wertpapiergeschäften über Homebanking eingeführt. HBCI verwendet für seine Dokumente ein auf das Privatkundengeschäft angepaßtes Format. Die Dokumente werden innerhalb des HBCI-Dialogs als abgeschlossene Einheit übertragen und enthalten Nutzdaten, die den eigentlichen Geschäftsvorfall beinhalten, sowie administrative Informationen, wie z.B. Dokumentennummern, eine eindeutige Referenznummer zur Kontrolle gegen Doppeleinreichung und eine *Digitale Signatur* zur eindeutigen Authentifizierung.
Im Gegensatz zu T-Online ist der multibankfähige ZKA-Standard unabhängig vom Bankinstiut und Kommunikationsmedium, d.h. dies ermöglicht den Einsatz über Internet, über „intelligente" Telefone (sog. Smartphones) oder Selbstbedienungsterminals.
HBCI schlägt zur Einhaltung von Datenintegrität, Verbindlichkeit von Aufträgen und zur Geheimhaltung von Dokumenten kein neues Sicherheitskonzept vor, sondern nutzt vorhandene Verfahren wie symmetrische Verschlüsselung nach dem Dreifach-DES-Standard (s. Anhang A), sowie die asymmetrische RSA-Verschlüsselung mit einer Schlüssellänge von 768 Bit.

Derzeit gibt es zwei Sicherheitslösungen beim HBCI:

DDV: Verwendung einer ZKA-Chipkarte (MAC) unter Verzicht auf den stärkeren RSA-Algorithmus (als Alternative wird Triple-DES verwendet). D.h. dies stellt eine Chipkartenlösung dar, die auf dem symmetrischen DES-Verfahren basiert. DDV verwendet einen MAC (s. *Kapitel 4*) als Signatur und verschlüsselt den Nachrichtenschlüssel mittels 2-Key-Triple-DES.

RDH: Dies stellt eine auf dem asymmetrischen RSA-Verfahren basierende Softwarelösung dar, welche zum Signieren und Verschlüsseln RSA verwendet.
Langfristig wird eine aufgrund erhöhter Sicherheit und besseren Bedienungskomforts RSA-basierte Lösung auf Basis einer vom ZKA zugelassenen Chipkarte angestrebt, welche laut neuester HBCI-Spezifikation 2.1 X.509-zertifizierte Schlüssel enthalten soll. Dies ist derzeit aufgrund technischer Restriktionen noch nicht flächendeckend umsetzbar.
HBCI 2.1 unterstützt deshalb noch Discetten bzw. die Festplatte als Speichermedium für den privaten Kundenschlüssel. Die Kryptofunktionen sind dann Teil der PC-Software des Homebankers.
Die gegenseitige Authentifikation zwischen Bank- und Kundensystem erfolgt während der Dialoginitialisierung durch Überprüfung der jeweiligen Digitalen Signatur, die später auch für die Integrität und Herkunft der Dokumente bürgen. Im Moment tragen die öffentlichen Schlüssel noch keine Zertifikate, d.h. eine Unterschrift eines Trust Centers. Mit der Zulassung der ersten Trust Center nach den Anforderungen des Signaturgesetzes zu Beginn dieses Jahres wird sich dies jedoch wahrscheinlich in absehbarer Zeit ändern. Unter HBCI findet die Generierung der Schlüsselpaare (geheimer und öffentlicher Schlüssel) jeweils bei der Bank bzw. dem Kunden statt. Von dem öffentlichen Schlüssel wird dann ein Hashwert erzeugt, der mit dem Schlüssel und einer handschriftlichen Unterschrift per Briefpost ausgetauscht wird. Bei Erhalt dieses Hashwerts wird dann der selbst berechnete Hashwert mit dem übermittelten verglichen. Stimmen sie überein, so wird der übermittelte Schlüssel als gültig akzeptiert. Gerade dieses Verfahren könnte dank zertifizierter Schlüssel digitalisiert und somit effizienter werden, indem man besagte Chipkartenlösung mit X.509-Zertifikaten anwendet.
Zur vertraulichen Übertragung von Finanzdaten wird jeweils ein aus einer Zufallszahl gebildeter Dreifach-DES-Schlüssel berechnet. Dieser wird mit dem öffentlichen Schlüssel des Verifizierers chiffriert und dem Dokument vorangestellt, das mit diesem Dreifach-DES-Schlüssel verschlüsselt wurde.
Als Ersatz für TANs werden beim Kunden und bei der Bank parallel Dialog- und Signaturzähler zum Schutz gegen wiederholtes Einreichen von Aufträgen geführt.
HBCI wird sich durch die starke Lobby der Finanzinstitute zumindest in Deutschland als Standard beim Homebanking etablieren. Die Bedeutung zertifizierter Digitaler Signaturen wird v.a. anhand der bisherigen, noch per Briefpost durchgeführten Schlüsselübermittlung deutlich. Die Übermittlung per Briefpost könnte

z.B. durch die zertifizierte Chipkartenlösung vermieden werden. Mit dem Einsatz Digitaler Signaturen könnte einiges an Übermittlungszeit und somit an Kosten eingespart werden ([Rae98], S.154ff).
Insgesamt kann man feststellen, daß HBCI auf dem rechten Wege ist, die Anforderungen des Digitalen Signaturgesetzes zu erfüllen. Allerdings bestehen derzeit noch Mängel, was die Zertifikatausgabe und v.a. Existenz von zertifizierten Schlüsseln angeht. Dies wird allerdings von den Entwicklern des HBCI-Standards auf lange Sicht hin durchaus angestrebt.

8.3.2 SET

VISA und Mastercard traten erstmals am 1. Februar 1996 mit der Idee eines von ihnen initiierten, technischen Sicherheitsstandard zum sicheren Einkaufen und Bezahlen per Kreditkarte im Internet an die Öffentlichkeit. Am 19. Dezember 1997 wurde durch die SETCo (Secure Eletronic Transaction LLC), die durch Visa und Mastercard (in voraussichtlicher Erweiterung mit JCB Credit Card Company und American Express) gegründete Allianz, die Implementierung dieses Standards beschlossen. SET Version 1.0 wurde am 31. Mai 1997 veröffentlicht. (Version 2.0 wird für Ende 1999 erwartet.) Dieser Standard nennt sich SET (Secure Electronic Transaction specification) und ist ein reines Zahlungsprotokoll. Es stellt für Finanzinstitute, Händler und Verkäufer eine neue Sicherheitsstufe für geschäftliche Transaktionen auf dem neuen Markt der Elektronischen Kaufhäuser und des sog. „Electronic Commerce" dar. Die SET-Spezifikation garantiert *Vertraulichkeit* und *Integrität* der übermittelten Daten, die *Authentizität* des Kartenbesitzers und des Händlers in der Weise, daß Transaktionen mittels einer *zertifizierten Zahlungskarte* in Verbindung mit einem entsprechenden Geldinstitut durchgeführt werden können.
Sie sieht eine hierarchische Zertifizierungsinfrastruktur und -ablauf vor, wie es in *Kapitel 6* geschildert wurde (und im Signaturgesetz vorgesehen ist). An der Wurzel befindet sich die sogenannte SET-Root-CA, welche der SETCo untersteht und als solche weitere CAs zertifiziert. Durch die Ausgabe digitaler Zertifikate an Besitzer einer Zahlungskarte ist garantiert, daß Transaktionen, die ein Kartenbesitzer „online" durchführt, auch gültig sind. Der Validierungsprozeß gilt auch für die an den Transaktionen teilnehmenden Banken, wodurch sich Sicherheit für sämtliche online-Transaktionen garantieren läßt.

Auch bei SET wird - wie beim HBCI - auf bestehende Sicherheitsmaßnahmen zurückgegriffen: Die symmetrische Verschlüsselung zur Kodierung der Zahlungsinformationen erfolgt nach dem DES-Algorithmus, zum Austausch der Sitzungsschlüssel werden RSA-Schlüsselpaare verwendet und die SHA-1-Funktion stellt mit einem 160-Bit-Hash die Integrität der übermittelten Dokumente sicher. Zertifikate werden im X.509-Format ausgegeben und sind nach dem DSA- oder RSA-Standard signiert.

SET sieht für jeden Karteninhaber ein Schlüsselpaar, für Banken und Händler je zwei Schlüsselpaare vor: eines zum Austausch der temporären Sitzungsschlüssel, das andere zum Unterschreiben der Dokumente. Für die öffentlichen Schlüssel wird ein gültiges Zertifikat benötigt, das von einer SET-CA-Hierarchie verankerten Zertifizierungsstelle beglaubigt sein muß.
SET ist ursprünglich ein Softwareprodukt, jedoch wird derzeit daran gearbeitet, SET-Hardware-Implementierungen für die Zertifizierungsstellen, die Server der Händler und die Finanzinstitute zu entwickeln. Eine SET-Transaktion besteht aus insgesamt acht Teilschritten, die den Zahlungsprozeß in entsprechende Dokumente abbilden.

Initialisierungsphase:
Die Bezahlung beginnt mit einer Anweisung (Purchase Initiate Request) des Karteninhabers an den Händler, den Kaufvorgang (engl. purchase) einzuleiten.
Der Händler sendet eine Rückmeldung (Purchase Initiate Response), die sein Zertifikat mit dem öffentlichen Chiffrierschlüssel zum Austausch des für diese Transaktion gültigen Sitzungsschlüssels zwischen Karteninhaber und Händler beinhaltet. Er unterzeichnet dieses Dokument mit seinem privaten Schlüssel, um zu gewährleisten, daß das Dokument nicht unbemerkt gefälscht werden kann. Damit der Karteninhaber die Unterschrift überprüfen kann, wird der entsprechende öffentliche Signaturschlüssel des Händlers in einem X.509-Zertifikat in der Rückantwort (Purchase Initiate Response) mitgeliefert.

Zahlungsphase:
Um mit der Zahlung zu beginnen, erstellt die SET-Software des Karteninhabers eine Kaufanfrage (Purchase Request), die die Auftragsabwicklung beim Händler anstößt. Daraufhin kann sich der Händler von der Auftragsbestätigung bei der für ihn zuständigen Abrechnungsstelle vergewissern, daß er für die bestellten Waren auch sein Geld erhält. Die Zahlungsinformation des Karteninhabers im „Purchase Request" setzt sich aus der Auftragsanweisung (Order Instruction) für den Händler mit einer Referenz auf die Bestellung sowie der Zahlungsanweisung (Payment Instruction) mit den Kreditkartendaten für die Abrechnungsstelle zusammen. Um Vertraulichkeit gewährleisten zu können, ist die Trennung dieser beiden Datensätze von höchster Wichtigkeit. Der Händler sollte nie die Kreditnummer zu Gesicht bekommen, sondern eine Bestätigung für deren Gültigkeit ist vollkommen ausreichend. Ebenso sind die Auftragsdaten für die Abrechnungsstelle bis auf die Kartennummer zum Feststellen der Gültigkeit der Kreditkarte geheimzuhalten. Ware soll jedoch nur dann ausgeliefert werden, wenn die mit ihr übermittelten Kreditkarteninformationen authentisch sind und von der Abrechnungsstelle als gültig bestätigt wurden. Um diese Trennung gewährleisten zu können, wurde für SET die sogenannte *Duale Signatur* entwickelt. (s.u.) Sie sorgt dafür, daß die Zahlungsanweisung vertraulich behandelt wird und die Bestellung der Abrechnungsstelle unzugänglich bleibt ([Rae98], S.142f).

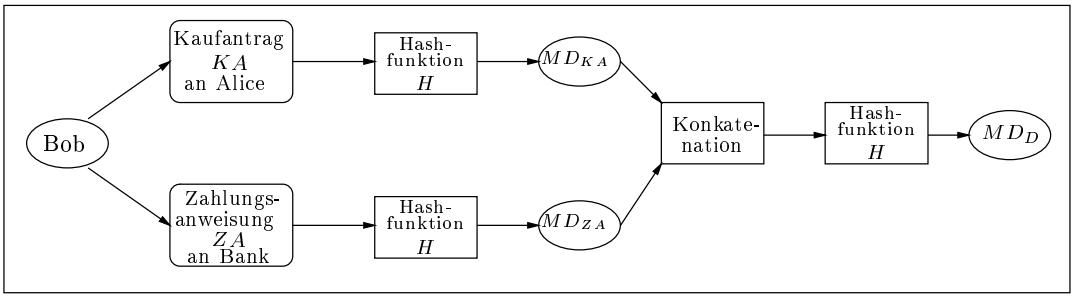


Abbildung 8.1: Erstellen einer Dualen Signatur bei SET

Digitale Signaturen:
Digitale Signaturen werden in SET mittels RSA unter Verwendung der 160-Bit-SHA-1-Hashfunktion, nach dem Verfahren: zuerst Hashwertbildung des zu signierenden Dokuments, dann Signieren des Hashwerts und anschließendes Verschlüsseln zur Datenübermitllung, wie unter *Kapitel 7* beschrieben, gebildet.
Duale Signaturen*:*
SET führt zudem eine neue Anwendung Digitaler Signaturen ein, die sogenannte duale Signatur. Um verstehen zu können, wozu man dieses neue Konzept benötigt, stellt man sich folgendes Szenario vor:
Bob möchte Alice ein Kaufangebot für einen Teil ihres Besitzes senden und dazu eine Vollmacht an seine Bank das Geld zu überweisen, falls Alice dieses Angebot akzeptiert. Dabei möchte Bob nicht, daß die Bank die Einzelheiten des Angebots, noch daß Alice Informationen über sein Konto erhält. Zudem soll das Angebot mit dem Geldtransfer so gekoppelt sein, daß Alice das Geld nur erhält, wenn Alice sein Angebot akzeptiert.
Er erreicht dies dadurch, daß er beide Nachrichten, das Kaufgebot an Alice und die Zahlungsanweisung an die Bank, mit einer speziellen Digitalen Signatur versieht, welche als duale Signatur folgendermaßen gebildet wird:
Generieren einer dualen Signatur:
Eine duale Digitale Signatur wird dadurch generiert, daß der Message Digest (s. *Kapitel 4*) des Kaufantrags MD_{KA} und der Message Digest der Zahlungsanweisung MD_{ZA} berechnet wird. Diese beiden Werte werden konkateniert und nochmals einer Hashfunktion unterzogen. Dieser berechnete Message Digest stellt sodann die *Duale Signatur* dar, s. Abb. 8.1. Dabei ist zu beachten, daß die Werte vor der Übermittlung jeweils verschlüsselt werden, um diese u.a. vor Fälschungen zu schützen.
Anwendungsprotokoll einer dualen Signatur:
Bob übermittelt die Zahlungsanweisung ZA, den Message Digest des Kaufantrags MD_{KA} sowie die duale Signatur MD_D an die Bank.
Alice erhält von Bob den Kaufantrag KA, den Message Digest der Zahlungsanweisung MD_{ZA} an die Bank und die zugehörige duale Signatur MD_D.

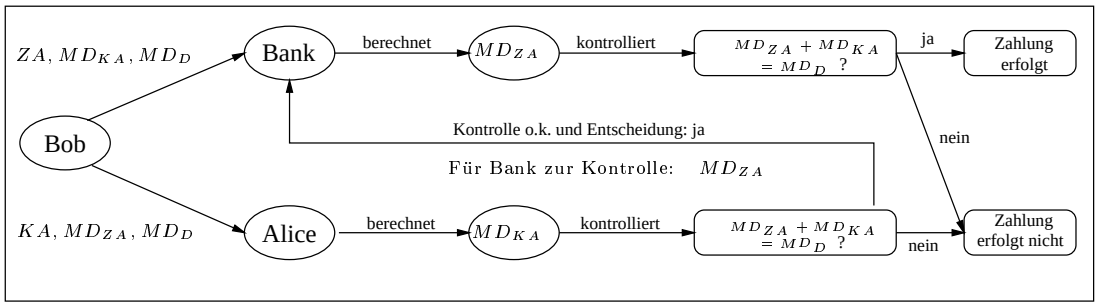


Abbildung 8.2: Anwendungsprotokoll einer Dualen Signatur bei SET

Akzeptiert sie das Angebot, so teilt sie das (Bob und) der Bank mit, indem sie der Bank als Beweis, daß der Kaufantrag an sie ging, den Message Digest der zugehörigen Zahlungsanweisung MD_{ZA} mitliefert.
Die Bank kann die Nachricht auf Authentizität überprüfen, indem sie den Message Digest zu der gesendeten Zahlungsanweisung MD_{ZA} berechnet, diesen Wert mit dem übermittelten Message Digest des Kaufantrags MD_{KA} konkateniert. Der anschließend berechnete Message Digest dieser Konkatenation muß dann identisch mit der Dualen Signatur MD_D sein. Die Duale Signatur mußte zuvor mit dem öffentlichen Verifikationsschlüssel auf Echtheit geprüft worden sein. Stimmen die Werte überein, so wird die Zahlungsanweisung an Alice von der Bank durchgeführt.
Somit kann die Bank die Authentizität des Angebots prüfen (s. Abb. 8.2), ohne das Angebot direkt sehen zu müssen ([SET97a], S.23f).

Vorteile von SET gegenüber gängigen Sicherheitsmethoden im Internet:
Der Hauptvorteil von SET ist, wie bisher vielleicht schon ersichtlich werden konnte, die Tatsache, daß mit digitalen Zertifikaten gearbeitet wird, anhand derer Kartenbesitzer und Händler mit den ihnen zugehörigen Finanzinstitut bzw. der Zahlungskarte eindeutig in Verbindung gebracht werden können. Digitale Zertifikate untermauern Geschäftsbeziehungen, indem sie Fälschung und Betrug in einem Maße verhindern, wie es bisherige Systeme noch nicht konnten. SSL (Secure Socket Layer), zum Beispiel, gewährleistet zwar Sicherheit bei der Übertragung von Daten im Internet, kann aber nicht für die Identität der beteiligten Transaktionspartner bürgen. Dies wird erst durch entsprechende Softwarekomponenten, wie sie durch SET bereitgestellt werden, ermöglicht.
Diese Zertifikate werden durch Banken an Händler und Kartenbesitzer (via Internet) angeboten. Eine CA gibt, wie bereits erwähnt, die entsprechenden Zertifikate elektronisch aus, nachdem vom Antragsteller die benötigten Informationen validiert werden konnten.

Zertifikate:
Auch bei SET dienen Zertifikate, der Authentifizierung der öffentlichen Schlüssel von Kartenbesitzern. Das *Signaturzertifikat* bindet implizit den öffentlichen Schlüssel an die primäre Accountnummer desselben (Primary account number: PAN), wobei die PAN durch ein kryptographisches Verfahren so verändert ist, daß nur die CCA (Cardholder Certification Authority), der Kartenbesitzer und der Herausgeber die Accountnummer kennen.
Der Kartenbesitzer übermittelt die Accountnummer und die geheime Variable der Rechnungsstelle, so daß dieser die Kartennummer trotz des verschlüsselten Wertes im Zertifikat des Kartenbesitzers verifizieren kann. Um die Vertraulichkeit des Kartenbesitzers nicht zu beeinträchtigen, wird der Name desselben nicht in das Zertifikat miteingeschlossen. Tatsächlich ist die veränderte Accountnummer ein Pseudonym des Kartenbesitzers.
Eine Funktion der Rechnungsstelle dient dazu, zu versichern, daß der geheime Schlüssel, der dazu verwendet wurde, die Zahlungsanweisung zu unterschreiben, auch der zum Konto gehörende Schlüssel ist. Damit nicht die PAN zu den entsprechenden Third Parties nochmals übermittelt werden muß, wird sie mittels einer Hashfunktion, die ebenso von einem Schlüssel abhängt, so verändert, daß der entsprechende Wert sorglos auf der Karte gespeichert werden kann. Die SET-Architektur erlaubt allerdings auch Kartenbesitzern ohne Signaturzertifikat an SET-Transaktionen teilzunehmen. Dies ist allerdings nur für Teilnehmer gedacht, deren Bankinstitut keine Zertifikate bereitstellt. Es liegt dann am Acquirer, inwieweit er solche Transaktionen zuläßt oder nicht.
Insgesamt erfüllt die Digitale Signatur bei SET die Bestimmungen des Digitalen Signaturgesetzes noch nicht hundertprozentig. Die Zertifizierung wird (noch nicht) von einer staatlich anerkannten Zertifizierungsstelle vorgenommen. Dies ist eine wesentliche Schwachstelle von SET, wie auch die Tatsache, daß nicht für jeden Teilnehmer ein Schlüssel zertifiziert ist. SET wird jedoch schon für viele Zahlungsanwendungen im Internet, z.B. in sog. elektronischen Kaufhäusern als Zahlungsprotokoll verwendet, was darauf schließen läßt, daß die Sicherheitsvorkehrungen in diese Richtung noch verbessert werden.

Abschließend kann man feststellen, daß zwar sowohl der HBCI-Standard, als auch das SET-Zahlungsprotokoll noch nicht die Anforderungen des Digitalen Signaturgesetzes erfüllen, jedoch die Richtlinien desselben für die Zukunft jedoch durchaus in Betracht ziehen.

Kapitel 9

Aktuelle Verläßlichkeit Digitaler Signaturen

„Eine Digitale Signatur kann als verläßlich gelten, wenn für den öffentlichen Signaturschlüssel, mit dem sie von jedermann überprüft werden kann, zum Zeitpunkt ihrer Erzeugung ein gültiges Zertifikat einer lizenzierten Zertifizierungsstelle bestand und wenn sie zum Zeitpunkt der Prüfung eine bestimmte Zeitdauer nicht überschritten hat oder andernfalls durch eine neue Signatur rechtzeitig konserviert wurde." ([GM97], S.415.)

- Haftung bei Verwendung Digitaler Signaturen

  „Mögliche Haftungsfragen sind aus den jeweiligen Verantwortlichkeiten und dem allgemeinen Haftungsrecht zu beantworten (jeder haftet für sein schuldhaftes Handeln oder Unterlassen)". ([GM97], S. 416) Dies bedeutet, daß die Haftung bei Mißbrauch Digitaler Signaturen abhängig vom jeweiligen „Trust Center" ist. Dies muß bei der Wahl der Zertifizierungsstelle mit berücksichtigt werden. [Ker98]

- Staatlich anerkannte Zertifizierungsstellen

  Die Regulierungsbehörde für Telekommunikation und Post ist das oberste deutsche Trust Center, das dazu berechtigt ist, andere Trust Center für Digitale Signaturen entsprechend den Anforderungen des deutschen Signaturgesetzes zu akkreditieren. [Ker98]
  Als erstes Unternehmen wurde bereits die Telekom zertifiziert, gute Chancen auf Genehmigung eines gesetzeskonformen „digitalen Notariats" hat das eigens gegründete „Joint-venture D-Trust" von debis und der Bundesdruckerei, aber auch die Commerzbank-Tochter „TC Trust-Center" und der Banknoten- und Chipkartenspezialist „Giesecke & Devrient".

- Einsatz Digitaler Signaturen

  Im Bereich des Vertragsabschlusses auf elektronischem Wege sind durch das Signatur-Gesetz noch nicht alle Schwierigkeiten beseitigt. Zum Teil ist für die Wirksamkeit eines Vertragsabschlusses , gesetzlich oder vertraglich, die eigenhändige Unterschrift bzw. die notarielle Beurkundung erforderlich. Das Bundesjustizministerium hat daher ein sog. Textformgesetz vorgeschlagen. Dadurch soll die elektronische Signatur der eigenhändigen Unterschrift gleichgesetzt werden. Erst dadurch würde es möglich, Rechtsgeschäfte, die der Schriftform unterliegen, durch Inanspruchnahme elektronischer Hilfsmittel wirksam abzuschließen. ([Pal98], S.29)
  Da die benötigte Sicherheitsinfrastruktur für den Einsatz Digitaler Signaturen bisher fehlt, sind diese zur Zeit nur innerhalb geschlossener Benutzergruppen im Einsatz. Der Sicherheitswert der auf dem Markt verfügbaren technischen Komponenten ist unterschiedlich und ohne eingehende Prüfung der Komponenten nicht hinreichend zu beurteilen. ([GM97], S.412)
  Seit Januar 1999 kann man nun von der Telekom eine Chipkarte für rund 50,-DM erwerben, mit der man mit dem auf ihr gespeicherten Geheimschlüssel Digitale Signaturen erzeugen kann. Diese „elektronischen Ausweise" sind bisher nur von der Telekom in ihren T-Punkt-Läden erhältlich und entsprechen dem Signaturgesetz. Zusammen mit einem passenden Chipkarten-Lesegerät und Spezialsoftware läßt sich eine Digitale Signatur unter Computerdokumente setzen. Damit eröffnet sich der Weg zum vollständigen Umstieg auf elektronische Schriftstücke, sei es nun um E-Mails, Interneteinkäufe oder Computersoftware, elektronische Arztrezepte, Verträge, Gerichtsurteile oder Steuererklärungen gesetzeskonform digital besiegeln zu können. Die verwendeten Schlüssel haben derzeit eine Länge von 1024 Bit und entsprechen somit den Empfehlungen des NIST.
  Als Signatur für die „breiten Massen" bietet die Telekom eine Erweiterung der Windows- Dateiverwaltung Explorer an, mit der sich beliebige Dateien unterzeichnen lassen. Dank einer Übereinkunft mit Microsoft arbeitet der Softwareriese bereits daran, seine populären Büroprogramme signaturkompatibel zu machen.
  Enorme Einsparungen werden v.a. im Bereich deutscher Behörden durch den Einsatz Digitaler Signaturen möglich. Nach einer Schätzung des Bundesarbeitsministeriums, können bis zu einer Milliarde Mark durch kürzere Übermittlungszeiten und geringeren Verwaltungsaufwand eingespart werden.

Jedoch wird es noch einige Jahre dauern, bis die *Digitale Signatur* salonfähig wird, da interessante Anwendungen für Normalbürger derzeit noch fehlen. Pilotanwendungen, wie z.B. das signaturgesicherte Abmelden der Biotonne in Karlsruhe scheint noch wenig attraktiv zu sein. [Weg99] Dennoch bahnt sich die Verbreitung Digitaler Signaturen bereits durch Verwendung deren Verwendung in vielen Softwarelösungen wie z.B. SET oder HBCI, sowie bereits bestehender Digitaler Signaturverfahren, wie PGP an.
Zudem eröffnen sich immer neue Einsatzgebiete Digitaler Signaturen, v.a. im Bereich des „Electronic-Commerce". Mit Hilfe von speziellen Signaturverfahren ist es möglich, „elektronisches Geld" im Internet anonym auszugeben, so daß kein Kaufprofil des „Internet-Einkäufers" erstellt werden kann. Der „Internetbummler" genießt somit die Vorteile des „Online-Shoppings" und muß dabei jedoch keinerlei Einbußen bzgl. seiner Privatssphäre in Kauf nehmen.

Letztendlich finden Digitale Signature in vielen Bereichen unseres Lebens ihre Daseinsberechtigung, sei es nun im Bankwesen, auf Behörden oder im Internet. Aufgrund der geschilderten Sicherheitsvorkehrungen Digitaler Signaturverfahren und der nun auch rechtlich, als auch organisatorisch und technischen geschaffenen Voraussetzungen in Form von gesetzlich anerkannten „Trust Centern", werden Digitale Signaturen einen raschen Einzug in unser tägliches Leben finden und alsbald gar nicht mehr wegzudenken sein.

Anhang A

Spezielle Algorithmen

A.1 *DES* (Data Encryption Standard)

Der *DES* stellt ein sehr häufig verwendetes Verschlüsselungsschema dar und wurde im Jahre 1977 durch das National Bureau of Standards (NBS), heute National Institute of Standards and Technology (NIST), unter der Bezeichnung „Federal Information Processing Standard 46" (FIPS PUB 46) als ein Verschüsselungsstandard in den USA angenommen.
Der *DES* verarbeitet Klartext, indem er jede 64 Bit lange Eingabe 16 Iterationen durchlaufen läßt, wobei am Ende jeder Iteration ein 64 Bit langes Zwischenergebnis erzeugt wird. Bei jeder Iteration werden die Bits permutiert und das Bitmuster durch ein anderes ersetzt. Die Eingabe jeder Iteration besteht aus der Ausgabe der vorhergehenden Iteration, wobei die Schlüsselbits zusätzlich einer Permutation unterworfen werden.
Der Vorgang der Entschlüsselung erfolgt analog der Verschlüsselung, wiederum in 16 Iterationen. Der verschlüsselte Text wird als Eingabe für den DES-Algorithmus verwendet, für die einzelnen Iterationen werden die während der Verschlüsselung erzeugten 16 Teilschlüssel in umgekehrter Reihenfolge verwendet.
Die Stärke von *DES* liegt v.a. darin, daß er Jahre intensivster Prüfung unbeschadet überstanden hat. Besorgnis ruft jedoch die Tatsache hervor, daß die Begründung für die Beschaffenheit des *DES-Algorithmus* geheimgehalten wird, da man durch diese Information evtl. Rückschlüsse auf Angriffsmöglichkeiten ziehen könnte. Außerdem ist die geringe Schlüssellänge von nur 56 Bit ein Unsicherheitsfaktor, da in einem 2^{56} großen Schlüsselraum mittels der Brute-Force-Methode (s. *Kapitel 2* u. *4*) im Durchschnitt 2^{55} Schlüssel zu testen sind, um den richtigen Schlüssel zu finden. ([Opp97], S.83)
Mittlerweile konnte auf der RSA-Daten-Sicherheits-Konferenz in San Jose in California gezeigt werden, daß der *DES* innerhalb von 22 Stunden und 15 Minuten zu brechen ist.

Dies gelang dem sog. „Distributed.Net", einer Gruppe von Computer-Enthusiasten, die über das Internet ein weltweites Netz von fast 100.000 PCs aufbauen konnten, unter Verwendung des „*DES*-Cracker" (dies ist ein von der EFF speziell dazu entwickelter Supercomputer) und der Brute-Force-Methode. Dieses weltweite Computer-Team konnte eine Nachricht entschlüsseln, welche mit dem *DES-Algorithmus* mit allgemein-verfügbarer Technologie und einem 56-Bit-Schlüssel chiffriert wurde.
„Distributed.Net" mußte dazu mit Hilfe des „*DES*-Cracker" von EFF 254 Billionen Schlüssel pro Sekunde testen, bis der zur Verschlüsselung verwendete Schlüssel gefunden werden konnte. Die Vorbereitungen für den Wettbewerb dauerten nur ein Jahr und die Kosten beliefen sich auf weniger als 250.000$.
Das erste Mal wurde der *DES* auf dem ersten RSA-Wettbewerb im Januar 1997 von einem Team aus Colorado, geleitet von „Rocke Verser of Loveland", in 96 Tagen gebrochen. Im Jahr darauf gelang es „Distributed.Net" bereits innerhalb von 41 Tagen. Diese Entwicklung zeigt den raschen Fortschritt der Technologie auf und ist zudem ein Hinweis darauf, daß *DES* seine Sicherheitsvorkehrungen verbessern muß. Mittlerweile wird von Experten eine *DES*-Schlüssellänge von mindestens 128 Bit empfohlen. [www98c]
Um den *DES* sicherer zu gestalten, wurde eine Alternative, der *Dreifach-DES*, entwickelt, der die Schwäche des *DES*: „des zu kurzen Schlüssels" umgeht.

A.2 *Dreifach-DES*

Der *Dreifach-DES* stellt eine Kombination aus Mehrfachverschlüsselung, *DES* und der Verwendung mehrerer Schlüssel dar. Dies zielt darauf ab, für mehr Sicherheit garantieren zu können und die Schwächen von *DES* auszugleichen.
Den *Dreifach-DES* gibt es in zwei Varianten: Mit zwei bzw. drei unterschiedlichen Schlüsseln und jeweils drei Durchläufen des *DES-Algorithmus*, wobei die Funktion einer Verschlüsselung-Entschlüsselung-Verschlüsselung (VEV-Folge) gehorcht.
Beim *Dreifach-DES* mit zwei Schlüsseln folgt die Chiffrierung der Gleichung:
Chiffretext $= V_{k_1}(E_{k_2}(V_{k_1}(\text{Klartext})))$ und die Dechiffrierung der Gleichung
Klartext $= E_{k_1}(V_{k_2}(E_{k_1}(\text{Chiffretext})))$.
Beim *Dreifach-DES* mit drei Schlüsseln folgt die Chiffrierung der Gleichung:
Chiffretext $= V_{k_1}(E_{k_2}(V_{k_3}(\text{Klartext})))$ und die Dechiffrierung der Gleichung
Klartext $= E_{k_3}(V_{k_2}(E_{k_1}(\text{Chiffretext})))$.
Durch die drei Iterationen der *DES-Funktion* beträgt die „effektive Schlüssellänge" 112 bzw. 168 Bit, so daß der Unsicherheitsfaktor des 56-Bit-langen Schlüssels beim einfachen *DES* behoben ist.
Anm.: Der *Dreifach-DES* ist zudem abwärtskompatibel, d.h. wenn man $k_1 = k_2$ bzw. $k_1 = k_2 = k_3$, so erhält man den ursprünglichen *DES-Algorithmus*.
([Sta95], S.236; [Opp97], S.86)

A.3 *IDEA* (International Data Encryption Algorithm)

IDEA ist ein blockorientierter konventioneller Verschlüsselungsalgorithmus, der im Jahre 1990 von Xuejia Lai und James Massey vom „Swiss Federal Institute of Technology" entwickelt wurde. Er verwendet einen 128 Bit langen Schlüssel, um Daten in Blöcken zu je 64 Bit zu verschlüsseln. Jeder Block wird in acht Durchläufen oder Iterationen, gefolgt von einer abschließenden Transformation verändert.
Der Algorithmus unterteilt die Eingabe in vier 16 Bit lange Teilblöcke. Jeder Iterationsdurchlauf nimmt vier 16 Bit lange Teilblöcke und erzeugt vier 16 Bit lange Ausgabeblöcke, die nach einer Transformation zur Bildung des 64 Bit langen verschlüsselten Textes verkettet werden.
Jede der acht Iterationen benutzt sechs der 16 Bit langen Teilschlüssel, die abschließende Transformation benutzt vier Teilschlüssel, was insgesamt 52 Teilschlüssel ergibt. Diese werden alle aus den ursprünglichen 128 Bit langen Schlüsseln erzeugt.
Jede Iteration von *IDEA* verwendet drei verschiedene mathematische Operationen, von denen jede auf zwei 16 Bit langen Eingaben ausgeführt wird, damit eine einzige 16 Bit lange Ausgabe entsteht. Diese Operationen sind:
Bitweises exklusives ODER
Ganzzahlige Addition *modulo* 216 (*modulo* 65536)
Ganzzahlige Multiplikation *modulo* $(2^{16} + 1)$ *modulo* 65537
Die Verwendung dieser drei voneinander unabhängigen Operationen als Kombination sorgt für eine komplexe Transformation der Eingabe, durch die eine Kryptoanalyse viel schwieriger wird als mit einem Algorithmus wie z.B. *DES*, der sich ausschließlich auf einfache Bit-Maskierung- und Permutationsfunktionen verläßt.
Die Stärke von *IDEA* liegt zum einen in der Schlüssellänge von 128 Bit, wodurch das System widerstandsfähiger gegen nicht-methodisches Vorgehen bei Angriffen ist und zum anderen darin, daß es schneller ist als der *Dreifach-DES-Mechanismus.*
([Sta95], S.237)

Literaturverzeichnis

[AB95] K.D. Wolfenstetter A. Beutelspacher, J. Schwenk. *Moderne Verfahren der Kryptographie*. Vieweg & Sohn Verlagsgesellschaft mbH, 1995.

[AM97] Scott Vanstone Alfred Menezes, Paul van Oorschot. *Handbook of applied cryptography*. CRC Press Inc., 1997.

[Beu93] Albrecht Beutelspacher. *Kryptographie*. Vieweg & Sohn Verlagsgesellschaft mbH, 1993.

[BI93] BI. *Duden der Informatik*. Dudenverlag, 1993.

[BSI97] BSI97. *Informationstechnik: Ende-zu-Ende-Sicherheit für elektronischen Dokumentenaustausch; Infrastruktur und Leitlinien für die Bundesverwaltung*. Informationsmaterial des Bundesamt für Sicherheit in der Informationstechnik, 1997.

[GM97] Andreas Pfitzmann (Hrsg.) Günter Müller. *Mehrseitige Sicherheit in der Kommunikationstechnik, Verfahren, Komponenten, Integration*. Addison Wesley, 1997.

[Ker98] PD Dr. Kersten. *Vortrag an der Universität Ulm: Digitale Signaturen (Juni 1998)*. Zertifizierungsstelle der debisIT Security Services, Bonn, 1998.

[Kö94] PD Dr. Johannes Köbler. *Skript zur Vorlesung WS 1993/94: Kryptographie*. Universität Ulm, 1994.

[KQN98] Yi Mu und Vijay Varadharajan Khanh Quoc Nguyen. A new digital cash scheme. In *Information Security: First International Workshop*, pages 313–320, California, 1998. Springer, Berlin-Heidelberg 1998.

[Mis98] Jean-Francois Misarsky. How (not) to design rsa signatures schemes. In *Public Key Cryptography: First International Workshop on Practice & Theory*, pages 16–20, Pacifico Yokohama, Japan, February 1998. Springer, Berlin-Heidelberg 1998.

[Opp97] Rolf Opplinger. *IT-Sicherheit: Grundlagen und Umsetzung in der Praxis.* Friedrich Vieweg & Sohn Verlagsgesellschaft mbH, 1997.

[Pal98] Christina Palm. Rechtliche Aspekte des elektronischen Geschäftsverkehrs. *Wirtschaft zwischen Alb und Bodensee: Das regionale Wirtschaftsmagazin der IHK Bodensee-Oberschwaben und Ulm*, 8:28–31, aug 1998.

[Pfi96] Birgit Pfitzmann. *Digital Signature Schemes: General Framework and Fail-Stop Signatures.* Springer-Verlag, Berlin, Heidelberg, New York, 1996.

[Rae98] Martin Raeplle. *Sicherheitskonzepte für das Internet.* dpunkt.verlag, 1998.

[Sch92] Prof. Dr. Uwe Schöning. *Theoretische Informatik kurz gefaßt.* BI-Wiss.-Verlag, 1992.

[Sch95] Prof. Dr. Uwe Schöning. *Skript zur Vorlesung WS1995/96: Algorithmen.* Universität Ulm, 1995.

[Sch96] Bruce Schneier. *Angewandte Kryptographie.* Addison Wesley, 1996.

[SET97a] SET (Secure Eletronic Transaction), http://setco.org. *Programmer's Guide-SET Secure Electronic Transaction Specification: SET97-Book1*, 1997.

[SET97b] SET (Secure Eletronic Transaction), http://setco.org. *Programmer's Guide-SET Secure Electronic Transaction Specification: SET97-Book2*, 1997.

[Sta95] William Stallings. *Datensicherheit mit PGP.* Prentice Hall Verlag GmbH, 1995.

[Sti95] D. R. Stinson. *Cryptography: Theory and Practice.* CRC Press, 1995.

[Th.92] R.L.Rivest Th.Cormen, Ch.E.Leiserson. *Introduction to Algorithms*, volume 7. MIT-Press, 1992.

[Weg99] Jochen Wegner. Elektronisches Siegel. *FOCUS*, 1:108–110, jan 1999.

[www98a] Deutsche Telekom AG, http://www.dtag.de. *Internetseite der Deutschen Telekom AG*, 1998.

[www98b] Institute for Telecooperation Technology, www.darstadt.gmd.de. *Internetseite über das MailTrust & TeleTrust Pilotprojekt: Digitale Signaturen für den Dokumentenaustausch*, 1998.

[www98c] *Internetseite über die Sicherheit von DES.* http://www.eff.org/descracker.html, 1998.

[www99a] SET (Secure Electronic Transaction), http://www.setco.org. *Internetseite über SET von VISA und Mastercard*, 1999.

[www99b] Sofitware, www.sofitware.com/Lexikon/E/Electronic-20Cash.htm. *Internetseite von Sofitware über Electronic-Cash*, 1999.

[www99c] *Internetseite über PEM (Privacy-Enhanced Mail).* http://www.cs.umbc.edu/ woodcock/cmsc482/proj1/pem.html, 1999.

Name: Silke Bergmaier Matr.Nr. 0278846

Erklärung

Ich erkläre, daß ich die Diplomarbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den 30. April 1999 .....