Kapitel 4

Grundlegende Sicherheitsmechanismen

Im Kapitel 4.1 werden zuerst einige grundlegende Sicherheitsmechanismen beschrieben, mit denen Kryptographiekonzepte für VPN-Systeme aufgebaut werden können. Anschließend erfolgt die Beschreibung der eigentlichen kryptographischen Algorithmen und der zur Verwaltung von Schlüsseln benötigten Infrastruktur.

4.1 Sicherheitsmechanismen für Verschlüsselung und Digitale Signatur

Dieses Kapitel erklärt Sicherheitsmechanismen, mit denen Daten auf ihrem Weg über öffentliche Netze geschützt werden können. Sicherheitsmechanismen sind die Werkzeuge, mit denen die jeweils erforderlichen Sicherheitsdienste wie Verschlüsselung und Digitale Signatur realisiert werden können.

Die Vertraulichkeit von Daten kann nur gewährleistet sein, wenn die Übertragung der Daten verschlüsselt erfolgt. Bei der Verschlüsselung gibt es verschiedene Verfahren (siehe auch /Rula94/).

4.1.1 Private-Key-Verfahren

Verschlüsselungsverfahren, die für die Verschlüsselung von Daten den gleichen Schlüssel verwenden wie für ihre Entschlüsselung, werden als symmetrische oder Private-Key-Verfahren bezeichnet.

Eines der bekanntesten, am weitesten verbreiteten und meistuntersuchten symmetrischen Verschlüsselungsverfahren ist der DES-Algorithmus, der 1978 in den USA normiert wurde (ANSI X3.92). DES steht für »Data Encryption Standard«. Der DES-Algorithmus wird heute meist als Triple-DES mit 112 Bit effektiver Schlüssellänge verwendet, wobei der DES-Algorithmus dreimal durchlaufen wird und zwar entweder mit zwei oder drei verschiedenen Schlüsseln (Abfolge A-B-A oder A-B-C). Bei dreimaligem Durchlauf mit je 56-Bit Schlüssellänge entspricht die Sicherheit des Verfahrens aber nicht etwa derjenigen einer einmaligen Verschlüsselung mit der Schlüssellänge von 3 * 56 Bit = 168 Bit, sondern ist geringer. Kryptologen haben deshalb den Begriff »effektive Schlüssellänge« eingeführt.

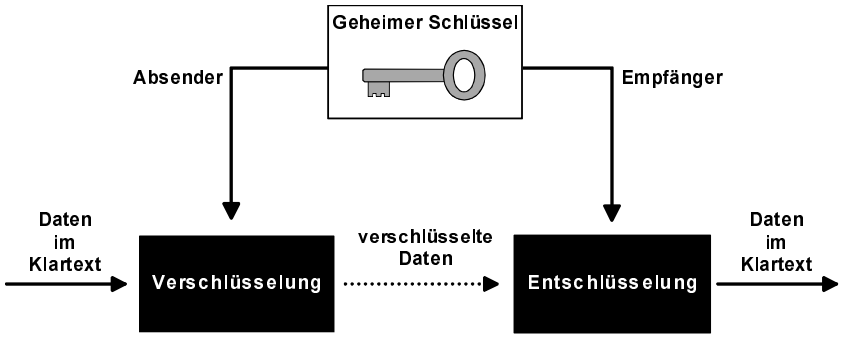

Abb. 4.1: Symmetrisches Verschlüsselungsverfahren (Private-Key-Verfahren)

Weitere symmetrische Verschlüsselungsverfahren sind beispielsweise IDEA (International Data Encryption Algorithm) und Safer (Secure And Fast Encryption Routine).

Zukünftig wird vermehrt der neue AES (Advanced Encryption Standard) verwendet, der mit einer Schlüssellänge von bis zu 256 Bit arbeitet.

Ein wesentlicher *Nachteil von Private-Key-Verfahren* (z.B. DES) ist, dass beide Kommunikationspartner über den gleichen Schlüssel verfügen müssen. Der Schlüssel, von dessen Geheimhaltung die Sicherheit abhängt, muss von einem Kommunikationspartner an den anderen übermittelt werden. Dies ist ein Unsicherheitsfaktor, dessen Risiko minimiert werden muss, indem man eine sichere Methode zur Schlüsselverteilung findet. Im Extremfall könnte dies die persönliche Übermittlung durch einen Kurier sein. Das Sicherheitsrisiko besteht darin, dass die Schlüssel, die geheimgehalten werden müssen, durch Nachlässigkeit, Vorsatz oder Zufall in falsche Hände geraten können.

Der *Vorteil von Private-Key-Verfahren* (z.B. DES) ist, dass sie sehr schnell sind. Es gibt zur Zeit schon Hardware-Lösungen, die bis zu 2,4 GBit/s und Software-Lösungen, die mehr als 100 MBit/s verschlüsseln.

4.1.2 Public-Key-Verfahren

Um das klassische Problem der Kryptographie, die Schlüsselverteilung, zu vereinfachen, wurden Verfahren entwickelt, die mit sogenannten öffentlichen Schlüsseln (Public Keys) arbeiten. Ein Public-Key-Verfahren oder asymmetrisches Verfahren arbeitet mit zwei verschiedenen Schlüsseln. Wird eine Verschlüsselung mit einem der beiden Teilschlüssel durchgeführt, kann nur mit dem dazu passenden Teilschlüssel die korrekte Entschlüsselung erfolgen.

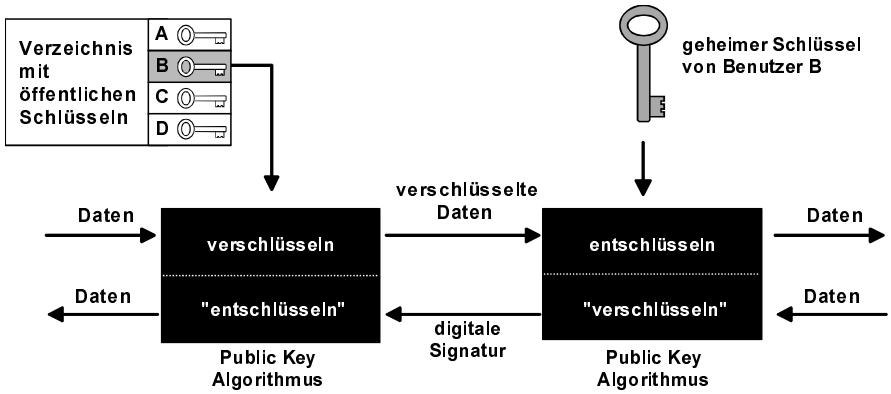

Abb. 4.2: Asymmetrisches Verschlüsselungsverfahren (Public-Key-Verfahren)

Aus der Kenntnis des einen Teilschlüssels kann der andere nicht berechnet werden. Aus diesem Grund kann ein Teilschlüssel ohne Bedenken veröffentlicht werden. Diesen bezeichnet man als »öffentlichen Schlüssel« (Public Key). Der andere Schlüssel muss geheimgehalten werden und heißt dementsprechend »geheimer Schlüssel« (Private Key) /Pohl90/.

Digitale Signatur

Eine wichtige Anwendung des Public-Key-Verfahrens ist die Digitale Signatur.

Daten, die mit einem bestimmten geheimen Schlüssel »verschlüsselt« wurden, können nur mit Hilfe des dazugehörigen öffentlichen Schlüssels wieder »entschlüsselt« werden (siehe Abb. 4.2). Hat nun eine Person die Daten mit ihrem geheimen Schlüssel digital signiert, kann mit Hilfe des öffentlichen Schlüssels überprüft werden, ob die digitale Signatur wirklich von dieser Person stammt.

Die erfolgreich durchgeführte Überprüfung ist der Beweis für die Authentizität der Signatur. Mit dem Prinzip der Digitalen Signatur steht somit ein Äquivalent zur handgeschriebenen Unterschrift zur Verfügung.

Das bekannteste Public-Key-Verfahren ist das RSA-Verfahren, mit dem gleichzeitig signiert und verschlüsselt werden kann (Das Kürzel RSA steht für die Entdecker Ron Rivest, Adi Shamir, Leonard Adleman).

Vertraulicher Austausch von Sicherheitsinformationen

Wird eine Information zuerst mit dem öffentlichen Schlüssel einer bestimmten Person verschlüsselt, kann diese Information nur von der Person, die den geheimen Schlüssel besitzt, rekonstruiert werden. Diese Anwendung erlaubt den vertraulichen Austausch sicherheitsrelevanter Informationen – zum Beispiel von Schlüsseln für symmetrische Verfahren wie DES.

Vorteile von Public-Key-Verfahren sind die Einsatzmöglichkeiten für ein einfaches Key-Management (Schlüsselverteilung) und für die Digitale Signatur.

Ein Nachteil von Public-Key-Verfahren ist, dass sie aufgrund ihrer Herkunft aus der Komplexitätstheorie sehr rechenaufwändig und deshalb nicht für die Verschlüsselung von großen Datenmengen geeignet sind.

4.1.3 One-Way-Hashfunktion

Die Digitale Signatur entspricht einer Operation mit dem Public-Key-Verfahren und ist daher sehr rechenintensiv.

Um den Aufwand zu vermindern, berechnet man nicht die gesamte Information mit dem Public-Key-Verfahren, sondern erstellt ein »Konzentrat« der Nachricht, das dann digital signiert wird.

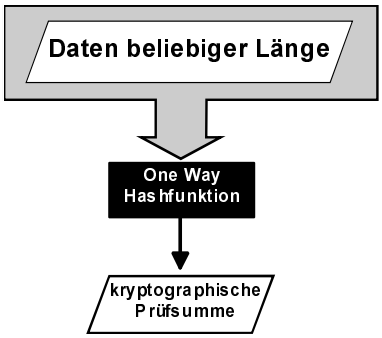

Abb. 4.3: One-Way-Hashfunktion

Auf eine Nachricht, deren Länge variabel ist, wird eine sogenannte One-Way-Hash-Funktion angewendet, die eine kryptographische Prüfsumme fester Länge als Ergebnis erzeugt. Zu den besonderen Eigenschaften von One-Way-Hash-Funktionen gehört, dass die Berechnung des Funktionswerts einfach ist, während es aber praktisch unmöglich ist, systematisch einen Wert zu finden, der dieselbe kryptographische Prüfsumme ergibt. Es ist also unmöglich, die Daten aus dem Hashwert zu ermitteln.

Eine kryptographische Prüfsumme muss eine Vielzahl von weiteren Eigenschaften aufweisen, die in ISO 10118 sowie in der Fachliteratur beschrieben sind. Verfahren in der Praxis sind unter anderem die One-Way-Hashfunktionen RIPEMD (EU-Projekt Réseaux IP Européens Message Digest), MD5 (Message Digest) und SHA-1 (Secure Hash Algorithmus).

4.1.4 Hybride Verschlüsselungstechnik

Da Public-Key-Verfahren wegen ihrer Komplexität zur Verschlüsselung von großen Datenmengen nicht geeignet sind und bei symmetrischen Verfahren die Schlüsselverteilung zu aufwändig ist, erweist sich eine Kombination von beiden als ideale Lösung: Die eigentliche Verschlüsselung der Daten eines Dokuments wird mit einem symmetrischen Verfahren (beispielsweise mit dem Triple-DES-Verfahren) durchgeführt, jedoch der Schlüssel für seine Verteilung mit dem Public-Key-Verfahren verschlüsselt wird. Eine Kombination des symmetrischen mit dem asymmetrischen Verschlüsselungsverfahren bietet neben der höchstmöglichen Sicherheit die Vorzüge der praktischen Handhabung.

4.1.5 Ein Wettlauf um die Sicherheit

Für die Sicherheit einer Verschlüsselung sind vier Faktoren ausschlaggebend:

- der verwendete Algorithmus,
- die Schlüsselgenerierung,
- die Schlüssellänge sowie
- die Aufbewahrung des Schlüssels.

Bei symmetrischen Verschlüsselungsverfahren geht man heute davon aus, dass die Praxissicherheit gegeben ist, wenn man eine Schlüssellänge von 128 Bit und mehr verwendet.

Eine vollständige Suche zur Entschlüsselung hieße, 2^{128} Schlüssel auszuprobieren. Damit stößt man auf ein praktisches Problem, denn mit den derzeitigen Ressourcen ist die Berechnung nicht in einer angemessenen Zeit möglich. Da aber die Geschwindigkeit von Computern, die man für diesen Angriff nutzen kann, immer weiter steigt, müssen in der Folge auch die Schlüssellängen von Zeit zu Zeit entsprechend vergrößert werden. Galt vor 10 Jahren noch eine praktische Sicherheit bei einer Schlüssellänge von 64 Bit als gegeben, so sind heute 128 Bit und in naher Zukunft 256 Bit erforderlich.

Gleiches gilt auch für Public-Key-Verfahren. In der Vergangenheit galt eine Schlüssellänge von 512 Bit als »sicher«, heute sind es 1024 Bit. Um langfristig Sicherheit zu gewährleisten, wird man zur Verwendung von 2048-Bit-Schlüsseln übergehen müssen.

Trotz aller Bemühungen gibt es keine absolute Sicherheit, da die Sicherheit anwendbarer Algorithmen mathematisch nicht bewiesen werden kann. Ein Algorithmus gilt dann als »sicher«, wenn fünf Jahre nach seiner Veröffentlichung die Mathematiker der Welt nicht in der Lage sind, ihn erfolgreich mathematisch anzugreifen. Nicht publizierte »geheime« Algorithmen gelten – weil nicht durch Experten überprüfbar – als »unsicher«.

4.1.6 Zertifizierungs-Systeme

Ein offenes Problem bei Public-Key-Verfahren ist die Frage, wie der öffentliche Schlüssel auf vertrauenswürdige Weise zum Kommunikationspartner gelangt. Selbst wenn man öffentliche Schlüssel verwendet, müssen diese authentisch ausgetauscht werden. Eine elegante Möglichkeit, öffentliche Schlüssel authentisch auszutauschen, ist die Einrichtung eines Zertifizierungs-Systems oder Trustcenters.

Der öffentliche Schlüssel jedes Benutzers wird dem Rechnersystem in Form eines Zertifikats von der vertrauenswürdigen dritten Instanz – der Zertifizierungs-Instanz – zur Verfügung gestellt. Dieses Sicherheitsprinzip ist im Directory-Authentication Framework /CCITT/ beschrieben.

Bei der Authentikation von Personen mit Hilfe von Ausweisen (Personalausweis, Reisepass usw.) fungieren die Behörden (Einwohnermeldeämter) als vertrauenswürdige dritte Instanz. Nachdem sich eine Person vorgestellt hat, kann man mit Hilfe eines Ausweises die Echtheit dieser Aussage verifizieren. Eine entsprechende Funktionalität muss zur Verfügung gestellt werden, um komplexe Sicherheitssysteme elektronisch zu realisieren.

Eine PKI stellt in aller Regel zentrale Sicherheitsdienste zur Verfügung, schafft also die Voraussetzungen dafür, daß eine Anwendung vertrauenswürdig realisiert werden kann.

Das folgende Bild zeigt im oberen Teil den prinzipiellen Aufbau einer Public-Key-Infrastruktur sowie einige Kommunikationskanäle. Im unteren Bereich ist schematisch eine Anwendung abgebildet, die auf der PKI-Grundfunktionalität basiert.

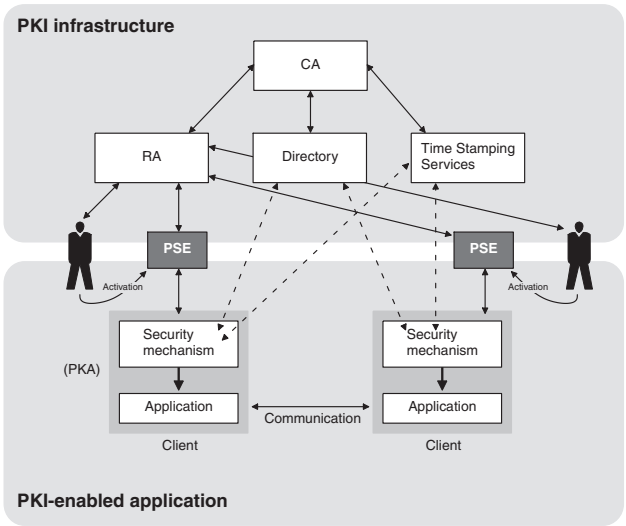

Abb. 4.4: PKI und PKA

Aufgaben und Komponenten einer PKI

Public-Key-Infrastrukturen bestehen aus Hardware, Software und einem abgestimmten Regelwerk, der Policy.

Die Policy definiert, nach welchen Sicherheitsregeln die Dienstleistungen erbracht werden. Dazu zählt das Betriebskonzept der PKI, die Benutzerrichtlinien sowie Organisations- und Arbeitsanweisungen.

Im Allgemeinen ist es üblich, die Registrierung der Teilnehmer und die Zertifizierung der Schlüssel voneinander zu trennen und zum Teil auch an unterschiedlichen Orten vorzunehmen.

Die **Registration Authority (RA)** kann als private (innerhalb einer Organisation) oder öffentliche Einrichtung betrieben werden. Ihre Hauptaufgabe besteht darin, die Anträge auf Zertifizierung zu erfassen und die Identität der Antragsteller entsprechend der Policy zu prüfen. Die Identitätsprüfung kann sehr einfach, z. B. per E-Mail, oder auch aufwendiger und sicherer, z. B. durch persönliches Erscheinen und Vorlage des Ausweises, erfolgen.

Die Registration Authority bildet die Schnittstelle zwischen den Teilnehmern bzw. Antragstellern und der **Certification Authority (CA),** an die sie die Anträge weiterleitet.

Die **Certification Authority** vergibt eindeutige Identitäten und verwaltet für jeden Teilnehmer ein oder mehrere Schlüsselpaare mit den dazugehörigen Zertifikaten. Jedes von der CA erzeugte Zertifikat verbindet den öffentlichen Schlüssel des Teilnehmers mit dessen Namen und zusätzlichen Daten (Gültigkeitszeitraum, Seriennummer, evtl. weitere Attribute).
Die Certification Authority gibt die Zertifikate aus und verwaltet sie, damit die öffentlichen Schlüssel und Attribute (Position im Unternehmen, Rechte usw.) der Teilnehmer möglichst einfach verifiziert werden können.

Zur Verwaltung der Zertifikate unterhält jede PKI einen **Directory Service**. Hier werden die gültigen zertifizierten öffentlichen Schlüssel der Teilnehmer veröffentlicht. Zurückgezogene oder kompromittierte Schlüssel werden in einer Sperrliste (**»Certificate Revocation List«, CRL**) zum Abruf bereitgehalten.

Ein **Zeitstempeldienst** dient dazu, gesicherte Zeitsignaturen gemäß der Policy zu erstellen. Damit wird ein Dokument oder eine Transaktion mit der aktuellen Zeitangabe verknüpft und diese Gesamtinformation anschließend digital signiert.

Das **Personal Security Environment (PSE)** ist die Sammlung aller sicherheitsrelevanten Daten eines Teilnehmers. Dazu gehören seine geheimen Schlüssel, die Zertifikate seiner Kommunikationspartner sowie der öffentliche Schlüssel der Zertifizierungsinstanz.

PKI-enabled Application

Als »PKI-enabled Application« (PKA) wird eine Anwendung bezeichnet, die auf der Grundlage der von der PKI zur Verfügung gestellten Sicherheitsdienste (Zertifikate, Verzeichnisdienst etc.) eine vertrauenswürdige Nutzung ermöglicht. Eine PKA enthält selbst unterschiedliche Sicherheitsmechanismen oder -verfahren (Authentisierung, Verschlüsselung etc.), mit denen Vertrauenswürdigkeit (Authentizität, Integrität, Verbindlichkeit, Einmaligkeit und Vertraulichkeit) erzielt wird.

Eine PKI bildet die Sicherheitsgrundlage für die vertrauenswürdige Nutzung von Anwendungen wie

- E-Mail
- Dokumentverschlüsselung (z. B. von MS-Office-Dokumenten)
- Transaktionen im Finanzbereich (EDIFACT)
- XML Prozessen
- SSL-Kommunikation
- VPN-Kommunikation
- Identifikations- und Authentisierungsprozessen
- Zahlungssystemen

Modelle von Public-Key-Infrastrukturen

Es gibt prinzipiell verschiedene Modelle von Public-Key-Infrastrukturen, die im folgenden kurz dargestellt werden:

- **Geschlossene Systeme**
  Eine Organisation betreibt eine PKI für eine oder mehrere Anwendungen (PKAs), die in ihrem eigenen Verantwortungsbereich liegen. Sicherheitsdienste wie z. B. gesicherte Kommunikation oder Authentisierung stehen nur innerhalb der Infrastruktur zur Verfügung.

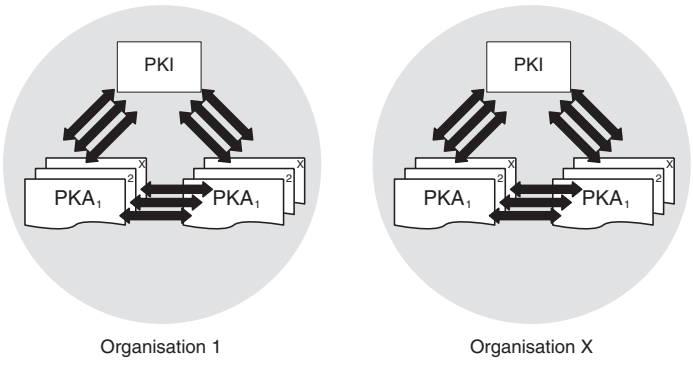

Abb. 4.5: Geschlossene PKI-Systeme

- **Offene Systeme**
  Mehrere Organisationen betreiben PKIs für eine oder mehrere Anwendungen, die in den Verantwortungsbereichen der unterschiedlichen Organisationen liegen. So ist z. B. die gesicherte Kommunikation zwischen den Organisationen möglich. Der Austausch beruht auf gegenseitigem Vertrauen sowie auf kompatiblen Technologien und Verfahren.

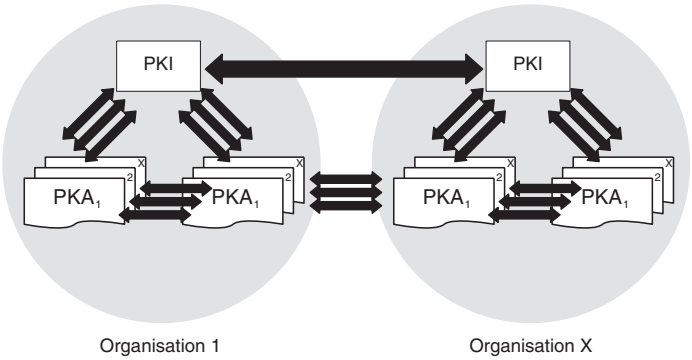

Abb. 4.6: Offene PKI-Systeme

- **Zentral administrierte Systeme**
  Ein PKI-Anbieter betreibt die PKI für eine oder mehrere Anwendungen, die in den Verantwortungsbereichen der sie nutzenden Organisationen liegen. Wenn die verschiedenen Organisationen der zentralen PKI vertrauen und kompatible Technologien und Verfahren verwendet werden, kann eine vertrauenswürdige Kommunikation realisiert werden.

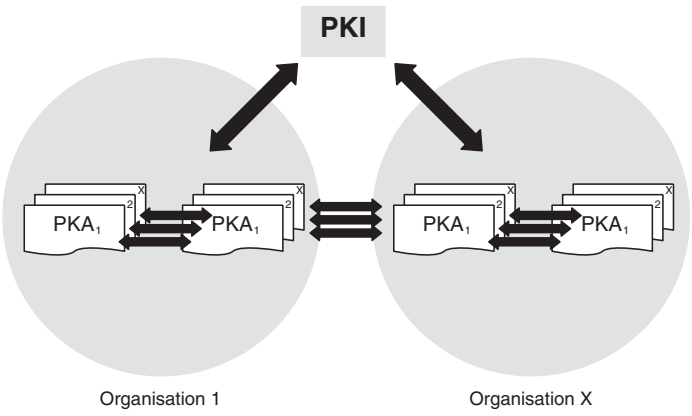

Abb. 4.7: Zentral administriertes PKI-System

Probleme mit PKIs in der Praxis

Bei der Nutzung von PKIs gab und gibt es einige Probleme, die im Folgenden diskutiert werden.

- Probleme bei geschlossenen Systemen
  »Geschlossenes System« bedeutet, dass die PKI nur innerhalb einer Organisation verwendet wird und nicht für die Kommunikation nach außen genutzt werden kann. Da jedoch in der Praxis viele organisationsübergreifende Prozesse stattfinden, ist der Nutzen einer solchen PKI sehr eingeschränkt.

- Probleme bei offenen Systemen
  Bei offenen Systemen muss zum Aufbau einer organisationsübergreifenden Kommunikation ein Abgleich der verschiedenen organisationsspezifischen Policies erfolgen. Ziel ist ein gemeinsames »Level of Trust«. Hier müssen geeignete Instrumente implementiert werden, um die organisatorischen sowie die IT-infrastrukturellen Konzeptionen zu bewerten, zu analysieren und zu gewichten.

  Gerade bei der Nutzung für personenbezogene organisationsübergreifende Prozesse stellt sich aus ökonomischer Sicht und aus den tatsächlichen Anforderungen heraus die Frage, ob das Signaturgesetz zwingend die Grundlage für die PKI und die zum Einsatz kommenden PKAs bilden muss. Hierbei muss berücksichtigt werden, dass viele organisationsübergreifende Prozesse automatisiert sind und somit nicht mehr personenbezogen arbeiten. Die Kardinalfrage in diesem Zusammenhang ist, ob innerhalb des Sicherheitskonzepts der PKI beispielsweise die Verantwortlichkeit für von Servern erstellte Signaturen geregelt ist (Haftungsausschluss).

  Hinzu kommt, dass eine Vielzahl von unterschiedlichen, teilweise sehr komplexen Standards existiert, die darüber hinaus der ständigen Weiterentwicklung unterliegen. Die Ursache hierfür liegt in der großen Vielfalt der Anwendungen (SSL, E-Mail etc.) und den daraus resultierenden besonderen Anforderungen.

- Unterschiedliche Verantwortung für PKIs und PKAs in Unternehmen
  Ein weiteres Problem, dem insbesondere große Organisationen gegenüberstehen, beruht darauf, dass die PKAs und PKIs zwar voneinander abhängig sind, aber häufig organisatorisch getrennt werden. In derartigen Fällen müssen sich beispielsweise verschiedene Abteilungen auf gemeinsame Ziele und Vorgehensweisen verständigen, um die entsprechenden technologischen Grundlagen zu erarbeiten.

- »Henne-Ei-Problem«
  Public-Key-Infrastrukturen sind nur dann ökonomisch sinnvoll, wenn der Einsatz dieser Strukturen und damit der vertrauenswürdige Ablauf von Geschäftsprozessen so umfassend wie möglich realisiert wird, d.h. wenn die gesicherte

Kommunikation mit so vielen Partnern wie möglich stattfinden kann. Voraussetzung dafür ist der konsequente Einsatz der bestehenden Technologien und die Umsetzung der Security Policies.

Die Realität ist aber, dass sich die beteiligten Organisationen nur schwer auf den Abgleich ihrer individuellen Sicherheitskonzepte einigen können. Dadurch gestaltet sich der Aufbau eines gemeinsamen »Level of Trust« langwierig und längst fällige Entscheidungen werden nicht getroffen. Zu viele Beteiligte nehmen noch eine abwartende Haltung ein und der Ausbau der bestehenden PKI-Infrastrukturen stagniert.

- Hoher personeller und organisatorischer Aufwand
  Die Einführung und der Betrieb einer Public-Key-Infrastruktur erfordert neben der technischen Umsetzung auch einen hohen personellen und organisatorischen Aufwand. Gerade in der Einführungsphase einer PKI ist die Sensibilisierung der Anwender für die IT-Sicherheit, die Schulung der Anwender auf die Produkte und die Planung und Durchführung des Roll-Outs ein nicht zu vernachlässigender Faktor.
- Key-Recovery bei der Verschlüsselung
  Falls Unternehmenswerte verschlüsselt werden, muss ein Verfahren realisiert werden, das bei technischen Defekten, bei einem PSE-Verlust oder beim Ausscheiden eines Mitarbeiters aus dem Unternehmen garantiert, dass die Unternehmenswerte sicher wieder entschlüsselt werden können.

Erstellung und Verifizierung von Zertifikaten

Der öffentliche Schlüssel eines jeden Benutzers wird dem System in Form eines Zertifikats zur Verfügung gestellt. Dieses enthält die Kennung der Zertifizierungsinstanz, die das Zertifikat erstellt hat, die Kennung des Benutzers, für den das Zertifikat erstellt wurde, den öffentlichen Schlüssel des Benutzers und eine Angabe zur Gültigkeitsdauer des Zertifikats (siehe Abb. 4.8).

Das Zertifikat ist von der Zertifizierungsinstanz, die es erstellt hat, digital signiert.

Jeder, der den öffentlichen Schlüssel der Zertifizierungsinstanz besitzt, ist damit in der Lage, zu überprüfen, ob der öffentliche Schlüssel eines Benutzers wirklich von der Zertifizierungsinstanz stammt.

Mit anderen Worten: Die Zertifizierungsinstanz veröffentlicht Zertifikate mit öffentlichen Schlüsseln, die die Zusammengehörigkeit von Benutzern und öffentlichen Schlüsseln bestätigen.

Nach dem Erhalt eines Zertifikats wird vom Sicherheitssystem die aktuelle kryptographische Prüfsumme über den Inhalt des Zertifikat berechnet. Außerdem wird aus der Signatur des Zertifikats und dem öffentlichen Schlüssel der Zertifizierungsinstanz unter Verwendung des Public-Key-Verfahrens die ursprüngliche kryptographische Prüfsumme berechnet.

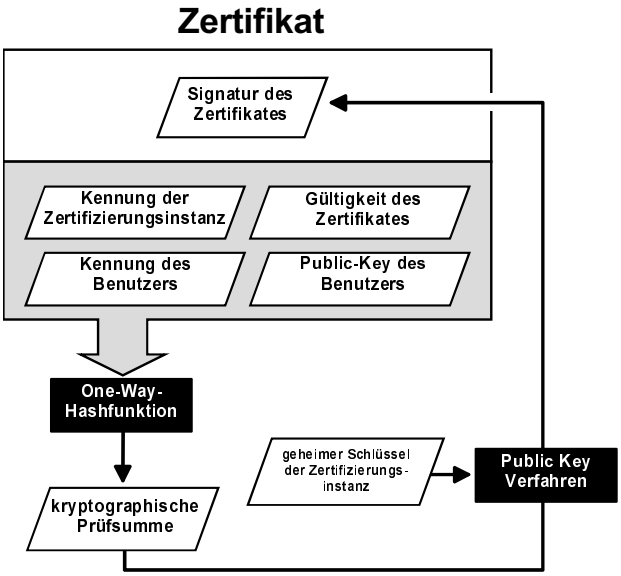

Abb. 4.8: Inhalt und Erstellung eines Zertifikats

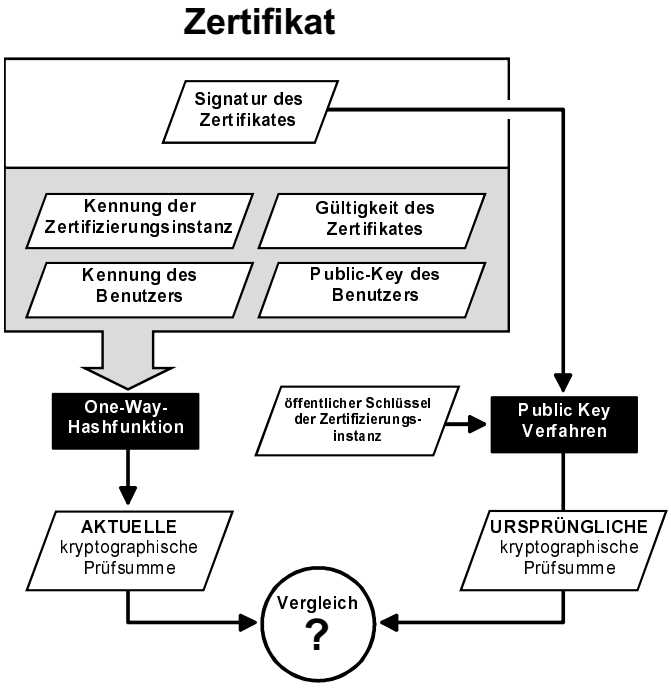

Abb. 4.9: Verifikation eines Zertifikats

Stimmen die beiden Prüfsummen überein, sind die Unversehrtheit und die Echtheit des öffentlichen Schlüssels des Benutzers, mit dem man eine Kommunikation durchführen möchte, bewiesen (siehe Abb. 4.9).

Voraussetzung ist, dass alle Benutzer des Sicherheitssystems der Zertifizierungsinstanz vertrauen können. Dazu muss die Zertifizierungsinstanz bestimmten Sicherheitsanforderungen genügen. Im Gesetz zur Digitalen Signatur werden die Sicherheitsanforderungen beschrieben, die eine Zertifizierungsinstanz erfüllen muss, die rechtlich anerkannte Zertifikate ausgeben möchte. Dazu zählen unter anderem vertrauenswürdiges Personal, zertifizierte Sicherheitskomponenten und eine vertrauenswürdige Systemumgebung. In einem globalen Sicherheitssystem können parallel oder hierarchisch verteilte Zertifizierungsinstanzen zusammengefasst sein.

Das hierarchische Schlüsselverteilverfahren ist die Grundlage für den Aufbau eines komplexen Sicherheitssystems.

4.1.7 Chipkarte (SmartCard)

Eine »intelligente Chipkarte« (»SmartCard«) ist ein Rechnersystem in der genormten Größe der EC-Karte (86 x 54 x 0,76 mm), das dem Benutzer Sicherheitsdienstleistungen zur Verfügung stellt.

Eine SmartCard enthält:

- eine CPU
- RAM- und ROM-Speicher
- ein »schlankes« Betriebssystem im ROM
- eine I/O-Schnittstelle, über die die gesamte Kommunikation stattfindet (Kontaktflächen oder kontaktloses Interface)
- ein EEPROM, auf dem die geheimen Schlüssel, zum Beispiel ein privater RSA-Schlüssel oder andere symmetrische Schlüssel, sowie persönliche Daten (Passworte etc.) sicher gespeichert sind
- sonstiges, beispielsweise einen Co-Prozessor, der symmetrische oder asymmetrische Verschlüsselung sehr schnell durchführt (Krypto-Prozessor)

Eine SmartCard stellt dem Benutzer in der Regel folgende Sicherheitsdienstleistungen zur Verfügung:

- Laden und Entladen von Werteinheiten für elektronisches Bezahlen (auch ohne Crypto-Prozessor)
- Kryptographische Anwendungen wie Digitale Signaturen usw.
- Identifikation/Authentikation des Benutzers (Aktivieren der Chipkarte)
- Single Sign On-Anwendungen (z. B. Passwort und PIN von unterschiedlichen Anwendungen)
- Lesen gespeicherter Servicedaten
- Sicheres Speichern von Daten auf der Chipkarte
- Ausführen sonstiger Rechenoperationen

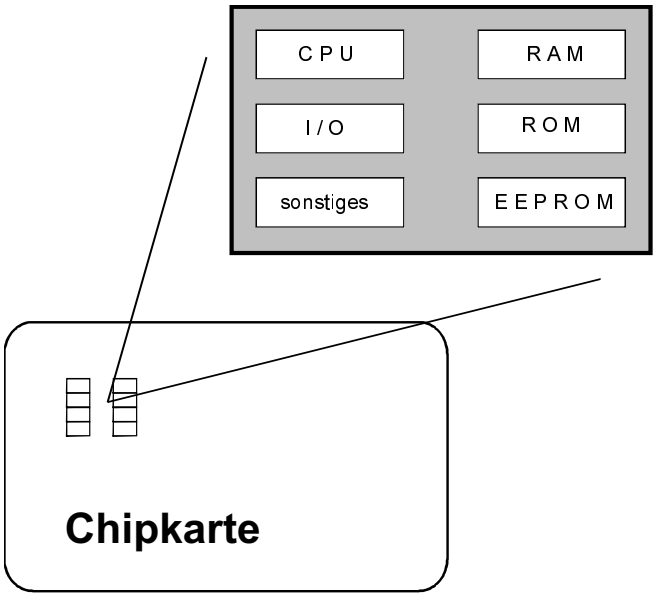

Abb. 4.10: Chipkarte

In Anwendungen wie dem Sicherheitssystem zum Schutz von elektronischen Dokumenten werden der öffentliche Schlüssel der Zertifizierungs-Instanz und der geheime Schlüsselteil des Benutzers gesichert in eine »intelligente Chipkarte« geladen. Die Chipkarte wird dann dem Benutzer vertrauenswürdig übergeben.

Aktivierung der Chipkarte

Die Chipkarte kann zum Beispiel durch ein Passwort geschützt werden. Wenn ein Benutzer des Sicherheitssystems Sicherheitsfunktionen in Anspruch nehmen will, muss er seine Chipkarte mit Hilfe seines persönlichen Passwortes aktivieren. Verliert der Benutzer seine Chipkarte, kann ein Finder diese nicht verwenden, da er das Passwort nicht kennt. Kennt jemand das individuelle Passwort eines anderen Benutzers, kann er keinen Nutzen daraus ziehen, wenn er nicht auch die Chipkarte besitzt. Außerdem kann ein Benutzer sein Passwort jederzeit ändern.

Sicherer als ein Passwort sind biometrische Identifikationsverfahren, die Körpermerkmale – zum Beispiel einen Fingerabdruck, die Stimme oder Gesichtszüge – zur eindeutigen Identifikation von Personen nutzen. Im Gegensatz zu einem Passwort kann ein solches Merkmal nicht gestohlen, verloren, vergessen oder weitergegeben werden.

Im englischsprachigen Bereich verwendet man häufig die Schlagworte »What you know« (Passwort), »What you have« (Chipkarte) und »What you are« (Biometrik) und fordert, bei einer Authentikation mindestens zwei dieser drei Verfahren zu kombinieren.

Multifunktionalität

Die Chipkarte ist so konzipiert, dass mit ihr mehrere Anwendungen möglich sind. Je nach Anwendung werden kontaktlose Chipkarten (z. B. für Zutrittssysteme) oder Chipkarten mit Kontakten verwendet.

Möglich ist zum Beispiel das Bezahlen an öffentlichen Telefonen oder an Point-of-Sales-Systemen (POS-Systemen), die Digitale Signatur von Dokumenten oder die Zugangskontrolle zu Gebäuden.

Mögliche Sicherheitsmechanismen einer SmartCard

SmartCard Hardware:

- Unter- und Überspannungsdetektion
- Erkennung niedriger Frequenzen
- gescramblete Busse
- Sensoren für Licht, Temperatur usw.
- Passivierungs- bzw. Metallisierungsschichten über Bus- und Speicherstrukturen oder über der gesamten CPU
- Zufallszahlengenerator in der Hardware
- spezielle CPU-Befehle für kryptographische Funktionen
- Speicherschutzfunktionen

SmartCard Software (z. B. Betriebssystem nach ISO 7816-4):

- Zugriffskontrolle auf Objekte
- Zustandsautomaten, die in Abhängigkeit von Identifikations- und Authentikationsmechanismen Befehle zulassen

Vorteile von SmartCards:

- Die kryptographischen Operationen werden auf der SmartCard ausgeführt. Der geheime Schlüssel verlässt die Karte niemals und kann somit nicht ausgelesen werden.
- SmartCards sind so klein wie Kredit- oder ec-Karten und können leicht überallhin mitgenommen werden.
- SmartCards sind flexibel für verschiedene Anwendungen einsetzbar.
- SmartCards sind mit Stückpreisen von 1,50 EUR bis 17,50 EUR (abhängig von der Stückzahl und dem Aufdruck) bedeutend preisgünstiger als andere Sicherheitsmodule.

Einsatzumfeld einer SmartCard

SmartCards werden typischerweise als Sicherheitskomponenten für Personen eingesetzt.

4.2 Kryptographische Algorithmen

Wie bereits erwähnt, nutzen VPNs eine Kombination von schnellen symmetrischen Verschlüsselungs-Verfahren bei der Online-Übertragung der Nutzdaten und langsamen asymmetrischen Verfahren zur Übertragung der geheimen symmetrischen Schlüssel. Nur so können die für eine Echtzeit-Übertragung erforderlichen Bandbreiten garantiert werden. Mit den zunehmenden Anforderungen bezüglich Schnelligkeit und Sicherheit wurden in der letzten Zeit vermehrt neue Verfahren entworfen, die VPN-Designer und -Entwickler in der nächsten Zeit beschäftigen werden.

Dabei ist die Länge der eingesetzten Schlüssel nur ein Kriterium, mit dem die Sicherheit eines Verfahrens bewertet werden kann. Ist das Verfahren nur mittels Brute-Force-Angriffen (Ausprobieren aller möglicher Schlüssel) zu knacken, steigt die Sicherheit exponentiell mit der Schlüssellänge. Das setzt aber ein ideales und fehlerfreies Verfahren voraus. Kann ein Algorithmus direkt gebrochen oder der Bereich für eine Brute-Force-Attacke eingeschränkt werden, tritt die Schlüssellänge als Sicherheitskriterium zurück. Verfahren, bei denen eine Entschlüsselung schneller als mittels Brute-Force vorgenommen werden kann, werden auch als »schwache Algorithmen« bezeichnet. Schwächen in Algorithmen zeigen sich manchmal erst nach Jahren, so dass eine hundertprozentige Sicherheit niemals garantiert werden kann.

4.2.1 Einführung

Ziel aller Krypto-Algorithmen ist es, eine unverschlüsselte Menge von Nutzdaten so zu manipulieren, dass der Besitzer eines passenden Schlüssels aus dem Ergebnis, den verschlüsselten Daten, den Klartext zurückgewinnen kann. Dabei wird zwischen zwei Klassen von symmetrischen Algorithmen unterschieden, den Block- und den Stromverschlüsselern.

Stromverschlüsseler

Stromverschlüsseler verschlüsseln den Klartext Bit für Bit mit einer Folge von generierten Bits, dem sogenannten Schlüsselstrom. In den meisten Fällen wird ein möglichst zufällig erzeugter Schlüsselstrom mittels einer XOR-Funktion mit dem Klartext verknüpft. Der Empfänger muss zur Entschlüsselung den gleichen Schlüsselstrom generieren können (Abb. 4.11). Ist der Schlüsselstrom eine Menge von echten Zufallszahlen, ist der Algorithmus theoretisch so abgesichert, dass Brute-Force die schnellste Angriffvariante ist.

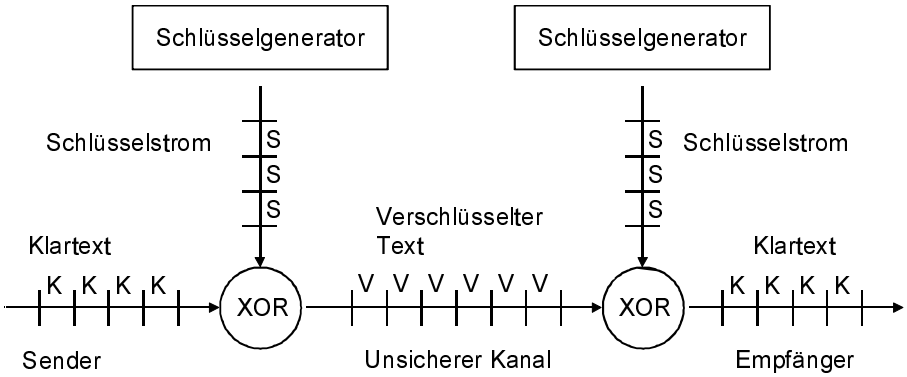

Abb. 4.11: Stromverschlüsseler

Entwickler von Stromverschlüsselern müssen sich mit dem Problem beschäftigen, wie der Schlüsselstrom des Senders zum Empfänger gelangen beziehungsweise von diesem neu berechnet werden kann. Geheimdienste verwenden oftmals Lochstreifen mit echten Zufallszahlen, die sich beim Sender und Empfänger befinden und nur ein einziges Mal benutzt werden können. Ein solches Verfahren wird auch als One-Time-PAD bezeichnet.

Da die externe Übertragung des Schlüsselstroms zum Empfänger bei der Implementierung von Online-Kommunikationsstrecken wie einem VPN nicht möglich ist, wurden Stromverschlüsseler entwickelt, bei denen Sender und Empfänger unabhängig voneinander auf beiden Seiten den Schlüsselstrom erzeugen. Dabei kann der Schlüsselstrom vom verschlüsselten Text abhängig sein oder nicht:

- Von einer synchronen Stromverschlüsselung wird gesprochen, wenn der Schlüsselstrom unabhängig vom verschlüsselten Text ist. Sender und Empfänger müssen dafür sorgen, dass der Schlüsselstrom auf beiden Seiten gleich ist (Abb. 4.12).

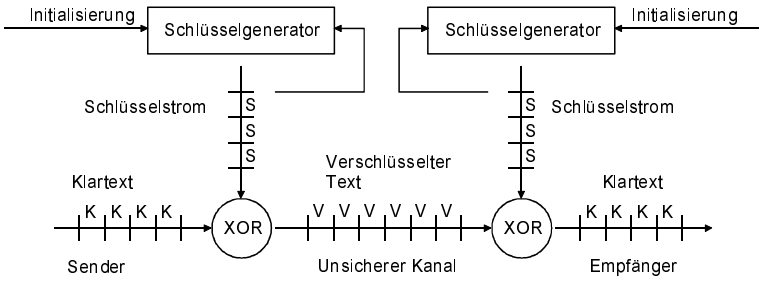

Abb. 4.12: synchroner Stromverschlüsseler

- Bei einer selbstsynchronisierenden Stromverschlüsselung ist der Schlüsselstrom eine Funktion des verschlüsselten Textes. Der interne Zustand des Algorithmus synchronisiert sich nach einer bestimmten Anzahl von entschlüsselten Bits automatisch und erzeugt anschließend den passenden Schlüsselstrom (Abb. 4.13).

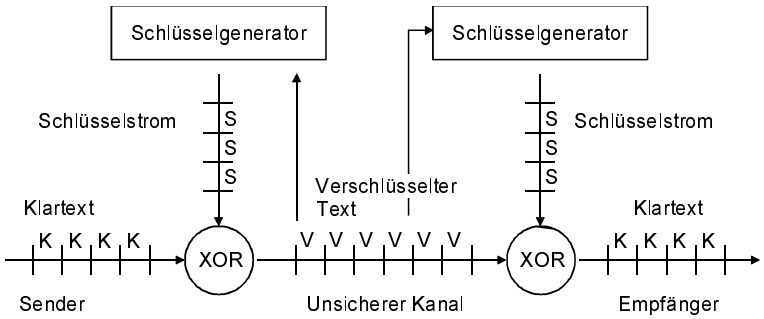

Abb. 4.13: selbstsynchronisierender Stromverschlüsseler

Blockverschlüsseler

Blockverschlüsseler zerlegen den Klartext in Blöcke konstanter Länge und erzeugen dann – abhängig vom Schlüssel – Block für Block den verschlüsselten Text. Falls der Klartext nicht exakt ein Vielfaches der Blocklänge ist, muss der letzte Block mit einem Bitmuster aufgefüllt werden (Padding). Blockverschlüsseler können ist verschiedenen Modi betrieben werden:

- Im ECB-Modus (Electronic Codebook) wird jeder Block unabhängig von den anderen verschlüsselt und gesendet (Abb. 4.14). Diese Methode birgt die Gefahr, dass ein Angreifer alte Blöcke mitschreibt und später wiederholen kann, ohne dass diese Manipulationen dem Empfänger auffällt (Replay-Angriff). Ein ECB-Verfahren kann nur komplette Blöcke verarbeiten, so dass Padding erforderlich ist.

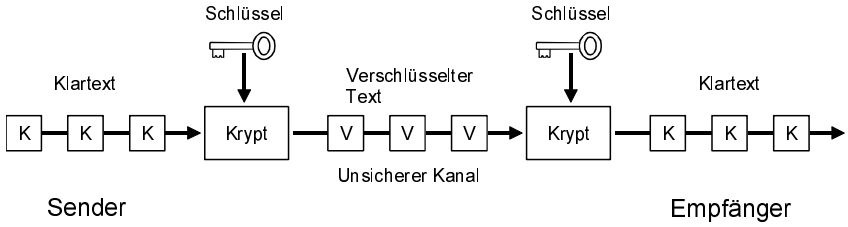

Abb. 4.14: Blockverschlüsseler im ECB-Modus

- Beim CBC-Modus (Cipher Block Chaining) findet eine Rückkopplung innerhalb des Verfahrens statt, indem ein Block des Klartextes mittels XOR mit dem letzten verschlüsselten Block verknüpft und anschließend verschlüsselt wird (Abb. 4.15). Das unbemerkte Einschleusen von Paketen ist jetzt nicht mehr möglich. Der erste Block des Klartextes wird dabei mit dem Ergebnis der Verschlüsselung einer Zufallszahl über XOR verknüpft. Diese Zufallszahl wird als Initialisierungsvektor IV bezeichnet und muss zusätzlich zum Schlüssel dem Empfänger der Nachricht übermittelt werden. Auch bei CBC können nur ganze Blöcke bearbeitet werden können.

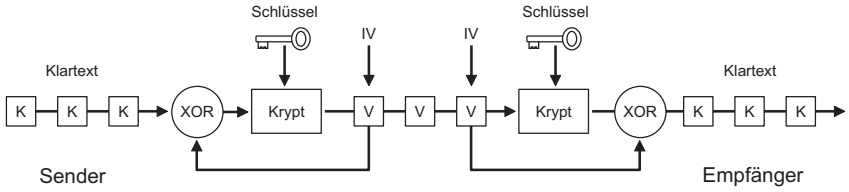

Abb. 4.15: Blockverschlüsseler im CBC-Modus

- In vielen Fällen ist es sinnvoll, aus einem Blockalgorithmus einen Stromverschlüsseler zu generieren. Das kann mit dem CFB-Modus (Cipher Feedback) geschehen, der aus einem Initialisierungsvektor IV und dem verschlüsselten Text den Schlüsselstrom erzeugt. Die zu verschlüsselnde Menge an Zeichen (meist ein Byte) wird mit dem letzten Byte des verschlüsselten IV mittels XOR verknüpft und auf die Reise geschickt. Dieses verschlüsselte Byte ersetzt zusätzlich ein einzelnes Byte des IV. Wenn acht Bytes verschlüsselt wurden und der IV ganz durch die verschlüsselten Daten ersetzt wurde, wird dieser neue Vektor verschlüsselt und steht für die nächsten XORs mit dem Klartext zur Verfügung usw. (Abb. 4.16). Da der Schlüsselstrom vom IV und den verschlüsselten Daten abhängt, handelt es sich um eine selbstsynchronisierende Stromverschlüsselung.

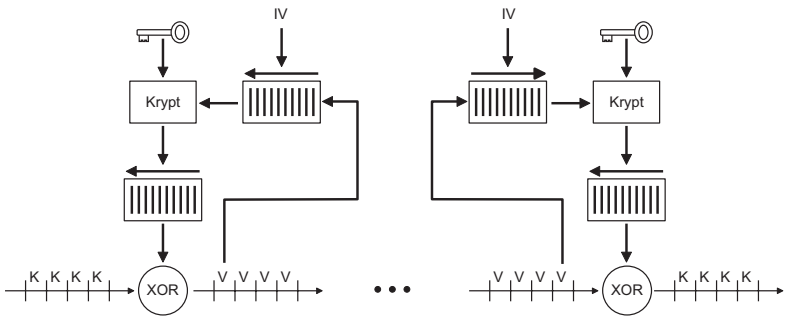

Abb. 4.16: Blockverschlüsseler im CFB-Modus

- Der OFB-Modus (Output Feedback) verläuft fast identisch zum CFB-Modus und macht ebenfalls aus einem Block- einen Stromverschlüsseler. Statt eines Bytes der verschlüsselten Daten wird jeweils das für das XOR mit den Nutzdaten benutzte Byte in das Schieberegister eingefüllt. Der Schlüsselstrom hängt nicht von Klartext oder dem verschlüsselten Text ab (Abb. 4.17), es handelt sich also um eine synchrone Stromverschlüsselung.

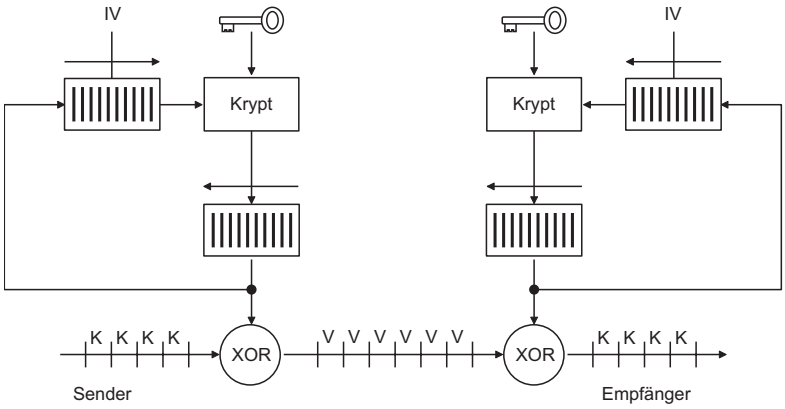

Abb. 4.17: Blockverschlüsseler im OFB-Modus

4.2.2 Symmetrische Verschlüsselungs-Verfahren

4.2.2.1 Data Encryption Standard (DES)

DES ist das wohl am weitesten verbreitete Verschlüsselungs-Verfahren. Obwohl bereits Mitte der siebziger Jahre von IBM entwickelt, konnte man ihm bis heute keine nennenswerten Schwächen nachweisen. Zahllose Applikationen auf der ganzen Welt setzen auf DES auf. In seiner ursprünglichen Implementierung hatte DES allerdings nur eine Schlüssellänge von 56 Bit, was gegen Brute-Force-Angriffe mit schneller paralleler Hardware heute als nicht mehr ausreichend gilt. Deshalb wurden Variationen von DES entwickelt, die eine größere Schlüssellänge bieten können.

Wegen der großen Verbreitung von DES gibt es zahlreiche Soft- und Hardware-Implementierungen, wobei spezielle Hardware eine verschlüsselte Übertragung von bis zu 1 GBit/Sekunde erlaubt. Die damit erzielbare Bandbreite genügt den Anforderungen an schnelle VPNs.

Beschreibung des Verfahrens

DES ist ein Blockverschlüsselungs-Verfahren, das immer ganze Datenblöcke von 64 Bit nimmt und jeden Block für sich verschlüsselt. Auch der Schlüssel hat eine Länge von 64 Bit, doch dient jedes achte Bit einer Paritätsprüfung und trägt nicht zur Verschlüsselung bei.

DES ist eine Folge von sogenannten Konfusionen und Diffusionen. Als Konfusion wird in der Kryptographie die Abbildung von Bitfolgen auf andere Bitfolgen bezeichnet. So könnte in einem trivialen Substitutions-Verfahren der Buchstabe a in den Buchstaben n, b in o usw. umgewandelt werden. Diffusion ist die Umstellung der Bits innerhalb einer Bitfolge, um das Knacken des Algorithmus mittels Häufigkeitsverteilungen zu erschweren. Eine triviale Diffusion ist die Permutation, bei der die Bits eines Blocks miteinander getauscht werden.

DES unterwirft die 64 Bit eines Blockes zunächst einer Eingangspermutation, bei der die Bits gemäß einer festen Tabelle miteinander vertauscht werden. Anschließend wird der Block in zwei Hälften zu je 32 Bit aufgeteilt. Die eigentliche Verschlüsselung besteht aus 16 »Runden«, in denen sich ebenfalls Permutationen, Substitutionen der Hälften und XOR-Operationen mit dem Runden-Schlüssel abwechseln. In jeder Runde Rx wird dabei ein anderer Anteil des Schlüssels genutzt, der aus dem Original-Schlüssel durch Permutationen erzeugt wurde. Abschließend werden die zuletzt entstandenen beiden Hälften mittels einer Schlusspermutation zu einem 64-Bit-Block kombiniert (Abb. 4.18).

Obwohl das Verfahren auf den ersten Blick etwas verwirrend erscheint, verläuft die Entschlüsselung exakt analog zur Verschlüsselung. Der verschlüsselte Text ist der Input jedes 64-Bit-Blocks, und bei Eingabe des bei der Verschlüsselung benutzten Schlüssels entsteht Block für Block der Klartext.

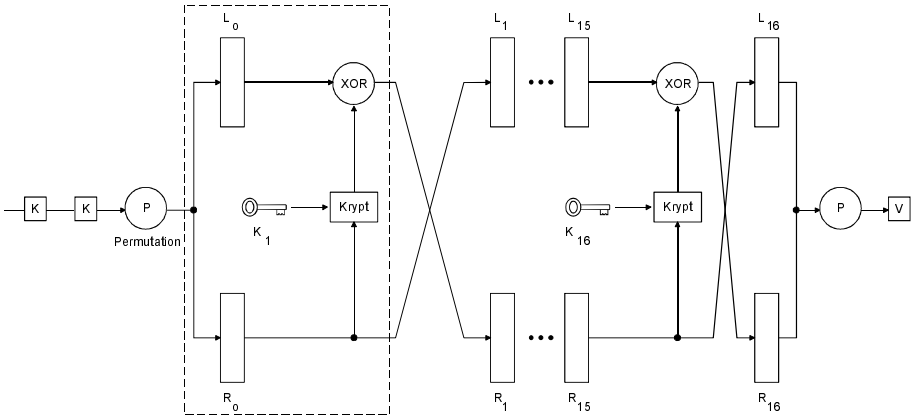

Abb. 4.18: Der DES-Algorithmus

Betriebsarten

Der DES-Algorithmus in der oben beschriebenen Form ist für viele Anwendungen nur begrenzt geeignet. Insbesondere für Terminal-Emulationen ist der Blockcharakter der Verschlüsselung ein Hindernis, da die zu übertragenden Daten in den seltensten Fällen exakt in die 64-Bit-Blöcke hinein passen. So muss gewartet werden, bis ein vollständiger Block verschlüsselt werden kann, oder es müssen unvollständige Blöcke gesendet werden. Bei vielen unvollständigen, etwa mit Nullen aufgefüllten Blöcken könnten sich Ansatzpunkte für Angreifer ergeben, da zumindest statistisch ein Teil des »Klartextes« (die Nullen) fest liegt. Um den Einsatz von DES für möglichst viele Anwendungen zu erlauben, wurden alle vier für Blockverfahren vorgesehene Betriebsarten definiert: ECB, CBC, CFB und OFB.

Bewertung der Sicherheit

Obwohl DES schon etwas in die Jahre gekommen ist, sind keine wirkungsvollen Angriffe gegen den Algorithmus selbst entwickelt worden. Die schnellste Methode des Knackens von DES-verschlüsselten Daten ist also der Brute-Force-Angriff, der gegen eine Schlüssellänge von nur 56 Bit allerdings vielversprechend erscheint. Deshalb wird DES in seiner klassischen Form heute nur noch selten eingesetzt. Mit Mehrfach-Verschlüsselung wie bei Triple-DES kann die Schlüssellänge erhöht werden. Bei höchstmöglichen Anforderungen an die Geschwindigkeit der Übertragung kann allerdings auf den klassischen DES zur Zeit nicht verzichtet werden.

Da der CBC-Modus einen guten Schutz vor eingefügten Blöcken darstellt und sich Bit-Manipulationen nur relativ gering auswirken, findet man DES (allerdings meist als Triple-DES) im CBC-Modus in vielen VPN-Implementierungen wieder.

4.2.2.2 Triple-DES

DES hat sich über Jahrzehnte hinweg als sicheres Verfahren erwiesen, einziger Nachteil ist die geringe Schlüssellänge. Deshalb schaltet man drei DES-Verschlüsselungen mit zwei oder drei Schlüsseln hintereinander, was natürlich zu einer deutlichen Verlangsamung der Ver- beziehungsweise Entschlüsselung führt. Die Verschlüsselung mit zwei Schlüsseln geschieht in der Reihenfolge A-B-A, die mit drei Schlüsseln in der Reihenfolge A-B-C (Abb. 4.19). Bei Triple-DES nach dem A-B-A Schema wird eine effektive Schlüssellänge von 112 Bit angenommen. Die effektive Schlüssellänge lässt sich mathematisch nicht geschlossen berechnen, so dass hier theoretische Überlegungen in den Vordergrund treten. Die oft zitierte effektive Schlüssellänge von 168 Bit bei einem Triple-DES nach dem Muster A-B-C ist in der Kryptographie umstritten, angenommen werden zwischen 112 und 168 Bit.

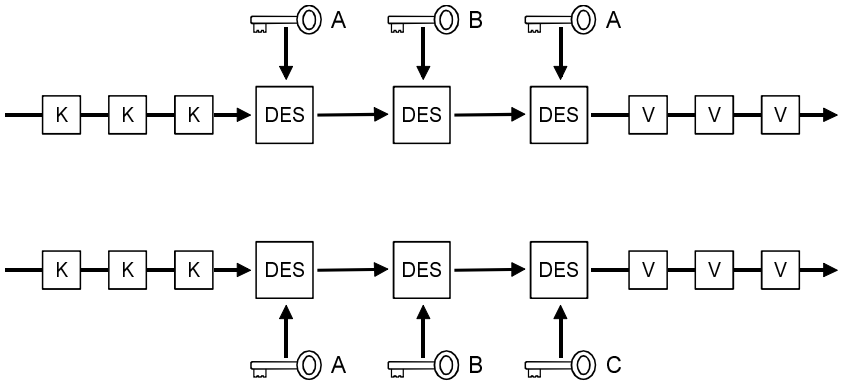

Abb. 4.19: Triple-DES in den Betriebsarten A-B-A und A-B-C

Auch Triple-DES lässt sich im CBC-Modus betreiben, wobei hier zwischen zwei Variantenunterschieden wird:

- Inner-CBC verschlüsselt die gesamte Datei dreimal hintereinander mittels DES-CBC. Bei dieser Methode sind drei Initialisierungsvektoren erforderlich (Abb. 4.20).

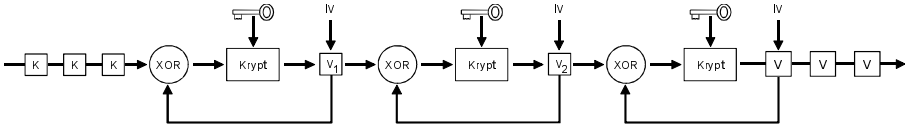

Abb. 4.20: Triple-DES im Inner-CBC-Modus

- Bei Outer-CBC durchläuft jedes Byte eine Dreifachverschlüsselung, bevor die Rückkopplung stattfindet (Abb. 4.21).

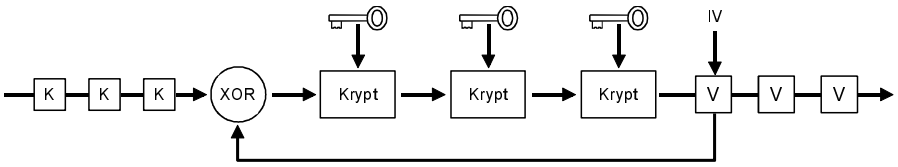

Abb. 4.21: Triple-DES im Outer-CBC-Modus

Analysen haben gezeigt, dass eine Mehrfachverschlüsselung nach dem Inner-CBC-Verfahren nur wenig mehr Sicherheit bietet als eine Einfachverschlüsselung, so dass in praktischen VPN-Implementierungen Triple-DES im Outer-CBC-Modus eingesetzt wird.

Da ein Verschlüsselungs-Vorgang mit Triple-DES exakt dreimal so lange dauert wie mit DES, kann in einem VPN mit Triple-DES nur eine Bandbreite von maximal einigen 100 KBit/s erzielt werden.

Der Vollständigkeit wegen sei auch der sogenannte Alternierende DES erwähnt. Er arbeitet so, dass er die gleichen Operationen wie der Single-DES durchführt, jedoch alternierend unterschiedliche Schlüsselsätze verwendet. Da sich somit der Schlüsselraum verdoppelt, erhöht sich die Sicherheit. Der Vorteil liegt darin, dass die Geschwindigkeit höher ist als bei einem längeren Schlüssel, der für vergleichbare Sicherheit notwendig wäre. (siehe auch www.isrc.qut.edu.au/paper.htm und eine Analyse in /Cart95/).

4.2.2.3 International Data Encryption Algorithm (IDEA)

Da das Sicherheitsbedürfnis der Europäer noch nie geringer war als das der Amerikaner, wurden auch an den europäischen Universitäten zahlreiche Projekte zur Entwicklung von kryptographischen Algorithmen durchgeführt. Eine der Triebfedern dabei waren die bis Anfang des Jahres 2000 bestehenden Exportrestriktionen für US-amerikanische Produkte, mit denen dann keine ausreichende Sicherheit mehr gewährleistet werden konnte. So hatte die Exportversion von DES nur eine Schlüssellänge von 40 Bit, die restlichen 16 Bit waren konstant und bei amerikanischen Stellen hinterlegt.

Eine dieser europäischen Entwicklungen ist der an der ETH Zürich Anfang der neunziger Jahre vorgestellte Algorithmus IDEA. Er ist allerdings patentiert, so dass bei kommerzieller Nutzung eine Lizenzgebühr an den Inhaber der Rechte (die Firma Ascom) gezahlt werden muss.

Für IDEA existieren bisher nur Software-Implementierungen, die aber immerhin doppelt so schnell sind wie Software-Versionen von DES.

Beschreibung des Verfahrens

IDEA ist ein Blockverschlüsselungs-Verfahren mit einer Blocklänge von 64 Bit und einer Schlüssellänge von 128 Bit. Zur Verschlüsselung werden Additionen, Multiplikationen und XOR-Verknüpfungen eingesetzt. Zunächst wird ein Datenblock in vier Teile zu je 16 Bit aufgespalten, die in insgesamt 8 Runden miteinander und mit sechs Teilschlüsseln der Länge 16 Bit verknüpft werden (Abb. 4.22). Um den Wertebereich dieser Teilblöcke nicht zu verlassen, werden Addition und Multiplikation modulo durchgeführt. Die Teilschlüssel werden aus dem Schlüssel durch Aufteilung und Shift-Operationen erzeugt. Die Entschlüsselung geschieht nach dem selben Verfahren, einzig die Teilschlüssel werden geringfügig anders berechnet.

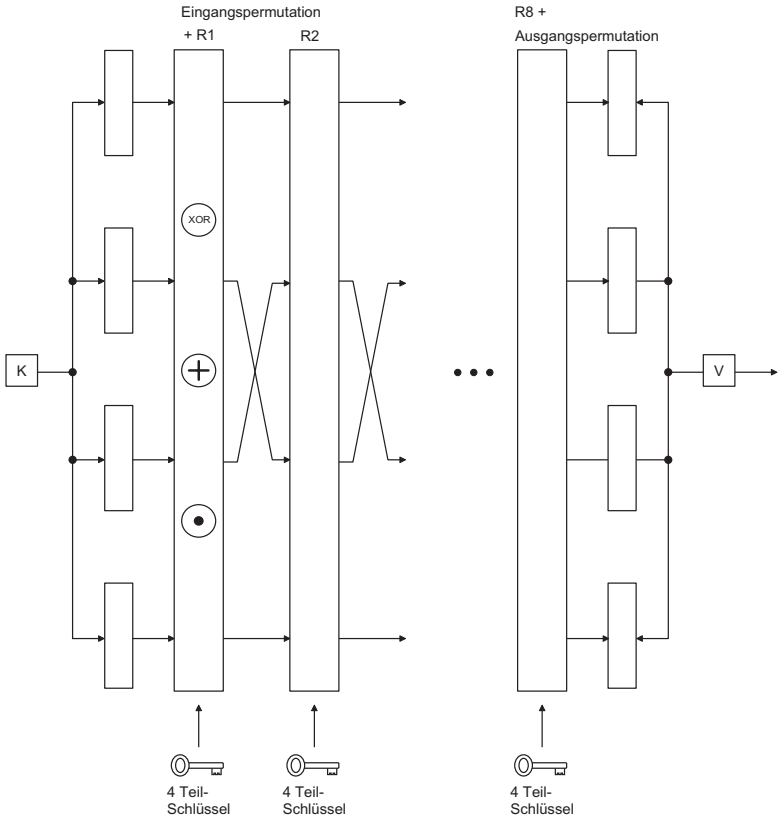

Abb. 4.22: IDEA

Betriebsarten

IDEA kann in allen Betriebsarten betrieben werden, in denen auch DES zu Hause ist: ECB, CBC, CFB und OFB. Theoretisch wäre sogar ein Triple-IDEA denkbar, das dann wegen der hohen effektiven Schlüssellänge auch von zukünftigen Hacker-Generationen mittels Brute-Force nicht anzugehen wäre. Für heutige Implementierungen ist die normale Schlüssellänge von 128 Bit allerdings ausreichend.

Bewertung der Sicherheit

Wegen der großen Schlüssellänge kann IDEA aus heutiger Sicht gegen Brute-Force-Angriffe als hoch angesehen werden. Es existieren einige schwache Schlüssel, bei denen ein »besserer« Angriff als Brute-Force möglich ist. Die Chance, einen solchen Schlüssel über einen Zufallsgenerator zu treffen, ist zum Glück verschwindend gering. IDEA ist allerdings noch nicht so gut erforscht wie DES. Es besteht also immer noch die theoretische Möglichkeit, dass eine Schwäche des

Algorithmus entdeckt wird, die Angriffe erlaubt, die um Größenordnungen effektiver sind als ein stupider Brute-Force-Angriff.

4.2.2.4 Blowfish

Das Blowfisch-Verfahren wurde 1993 vom Kryptographie-Experten Bruce Schneier entwickelt. Es handelt sich ebenfalls um eine Blockverschlüsselung mit 64 Bit-Blöcken, die Schlüssellänge ist variabel und kann bis zu 448 Bit reichen. Da Blowfish zudem frei verfügbar ist und auch von seiner Geschwindigkeit Vorteile bietet (die Software-Version ist schneller als die DES-Software-Version), hat es schnell eine beachtliche Marktpräsenz gewinnen können.

Beschreibung des Verfahrens

Die innerhalb von Blowfish verwendeten mathematischen Operationen sind auf Einfachheit hin ausgewählt; es werden Additionen, Index-Operationen und XOR-Verknüpfungen eingesetzt. Der Algorithmus besteht aus 16 Runden, in denen der in zwei Teilblöcke von je 32 Bit zerlegte 64-Bit-Block einer Permutation und einer Substitution unterworfen wird. Diese Funktionen sind von insgesamt 18 Teilschlüsseln abhängig, die aus dem vorgegebenen Schlüssel durch eine Expansionsfunktion berechnet werden (Abb. 4.23). Eine Software-Implementierung von Blowfish auf einem 32-Bit-Prozessor ist schneller als DES. Leider braucht das Verfahren recht viel Cache-Speicherplatz, so dass eine Implementierung auf Chipkarten etc. nicht möglich ist.

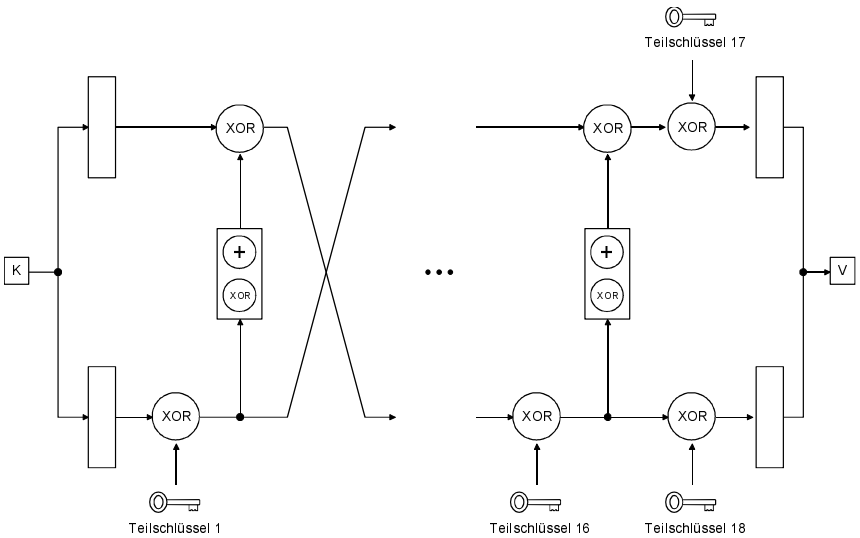

Abb. 4.23: Blowfish

Bewertung der Sicherheit

Abgesehen von eingigen schwachen Schlüsseln ist bisher kein Makel am Blowfish-Algorithmus bekannt. Wird die maximal mögliche Schlüssellänge von 448 Bit gewählt, ist ein Brute-Force-Angriff auch in der mittelfristigen Zukunft aussichtslos. Der aus Blowfish abgeleitete Algorithmus Twofish unterlag im Wettstreit im den neuen AES-Standard allerdings klar seinem Konkurrenten Rijandael.

4.2.2.5 RC4 und RC5

RC4 ist eine synchrone Stromchiffrierung mit einer variablen Schlüssellänge von bis zu 128 Bit, bei der der Klartext Byte für Byte mit einer Pseudo-Zufallszahl über ein XOR verknüpft ist. Der Algorithmus wurde schon 1987 von Ron Rivest entwickelt und lange Jahre geheim gehalten. Nachdem der Quellcode jedoch im Internet veröffentlicht wurde, konnten sich Kryptologen endlich ein Bild von der Qualität des Verfahrens machen. RC4 ist geschützt, die Rechte liegen bei der US-amerikanischen Firma RSA-Security. RC5 ist eine Weiterentwicklung des RC4-Verfahrens, die hier jedoch nicht näher behandelt wird.

Beschreibung des Verfahrens

Der Schlüsselstrom für RC4 wird aus einem Feld mit 8 * 8 = 64 Bytes erstellt, das zunächst mit einer Untermenge der Zahlen von 0 bis 255 gefüllt wird. Auswahl und Reihenfolge der Zahlen hängen vom Schlüssel ab. Aus diesem Feld wird dann der Schlüsselstrom über einige Additionen und eine Tauschoperation erzeugt (Abb. 4.24). Durch diese recht einfache Berechnung ist der Algorithmus sehr schnell, etwa fünf bis zehn mal so schnell wie DES.

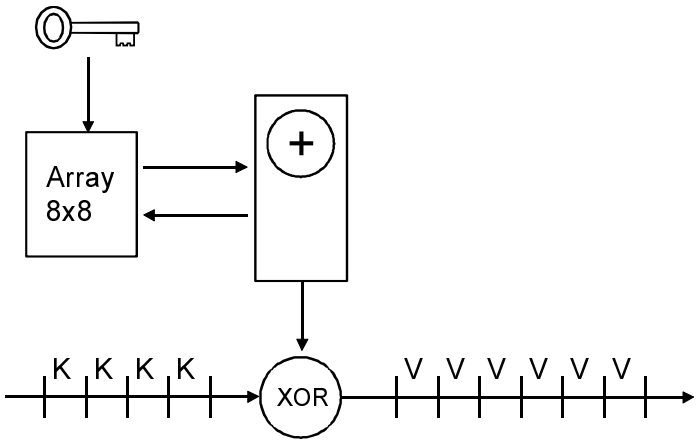

Abb. 4.24: RC4

Bewertung der Sicherheit

RC4 kommt in zahlreichen kommerziellen Produkten zum Einsatz. Deshalb ist es schon fast katastrophal, dass der Algorithmus sich als sehr schwach erwiesen hat (siehe /Fluh01/ und /Stubb01/). Es existieren viele Schlüssel, bei denen man dem verschlüsselten Text schon mit einer gewissen Wahrscheinlichkeit ansehen kann, wie der geheime Schlüssel aussehen könnte. Bei genügend vielen abgehörten Netzwerkpaketen ist das Knacken der RC4-Kommunikation dann möglich. RC4 sollte unter keinen Umständen mehr eingesetzt werden.

4.2.2.6 Advanced Encryption Standard (AES)

Die Entwickler von VPNs müssen oft Kompromisse zwischen Schnelligkeit und Sicherheit eingehen. Das schnellste der klassischen Systeme, Hardware-DES, hat wegen seiner geringen Schlüssellänge nach heutigen Maßstäben nur eine relativ geringe Sicherheit. Der Einsatz größerer Schlüssellängen führt zu Einbußen bei der Geschwindigkeit.

Das US-amerikanische NIST (National Institute of Standards and Technology) begann deshalb 1997 mit der Suche nach einem Nachfolger für DES, der »AES« (Advanced Encryption Standard) genannt wurde. In einer Ausschreibung wurden bis August 1998 fünfzehn Algorithmen präsentiert, die Kryptologen und anderen Interessenten zur Untersuchung übergeben wurden.

Die Verfahren wurden nach den Kriterien Sicherheit, Kosten (Speicher, Prozessorlast) sowie der Charakteristik von Algorithmen und möglichen Implementierungen untersucht. Am 2. Oktober 2000 wurde der von den belgischen Kryptologen Joan Daemen und Vincent Rijmen (Universität Leuven) entwickelte Algorithmus Rijndael zum Sieger erklärt. Rijndael ist frei von Patenten, so dass seiner raschen Verbreitung nichts im Wege steht. Der Algorithmus kann einfach auf verschiedenen Prozessor-Typen (8-Bit oder 32-Bit) implementiert werden. Da interne Tabellen direkt in der Hardware verdrahtet werden können und zudem bestimmte Operationen parallel ausgeführt werden können, ist die Entwicklung schneller Hard- und Software-Varianten nur eine Frage der Zeit.

Rijndael ist, ähnlich wie DES, ein Blockverschlüsselungs-Verfahren, welches aus mehreren Runden besteht. Die Länge der zu verschlüsselnden Blöcke kann 128, 192 oder 256 Bit betragen, ebenso wie die Länge des Schlüssels. Wird eine Schlüssellänge von 256 Bit gewählt, ist eine erfolgreiche Brute-Force-Attacke wegen der immerhin 2^{256} Möglichkeiten auf Jahrzehnte hinaus praktisch ausgeschlossen. Die Anzahl der Runden hängt von der Blocklänge und der Schlüssellänge ab und beträgt 10, 12 oder 14 (Tab. 4.1).

AES-Runden	Nb = 4	Nb = 6	Nb = 8
Ns = 4	10	12	14
Ns = 6	12	12	14
Ns = 8	14	14	14

Tabelle 4.1: Anzahl der AES-Runden in Anhängigkeit von der Blocklänge Nb und der Schlüssellänge Ns (in Bytes)

Jede der Runden von Rijndael besteht aus einer Reihe von Byte-orientierten Transformationen, in denen die Autoren die Stärken vieler anderer Verschlüsselungs-Algorithmen kombiniert haben. Die eingesetzten Operationen haben sich bei anderen Verschlüsselungs-Verfahren in der Vergangenheit als widerstandfähig gegenüber Angriffen erwiesen.

Die in einem zweidimensionalen Array abgelegten Zeichen des Klartext-Blocks werden zunächst der sogenannten ByteSub-Transformation unterworfen. Es handelt sich um eine nichtlineare Substitution der einzelnen Bytes, die über eine Tabelle (S-Box) festgelegt wird. Abbildung 4.25 zeigt die Transformation für den Fall einer Blocklänge von 192 Bit, bei denen der Block in einem Array von 6x4 Bytes abgelegt ist.

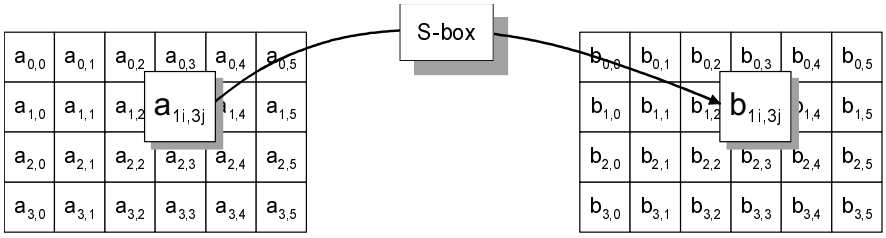

Abb. 4.25: ByteSub-Transformation

Die Bytes werden anschließend der ShiftRow-Transformation unterworfen, bei der die Zeilen des Arrays bis auf die erste zyklisch geshiftet werden, jede Zeile um eine andere Anzahl von Bytes. Abb. 4.26 zeigt wieder den Fall der Blocklänge von 192 Bit.

Die MixColumn-Transformation unterwirft jede Spalte des Arrays einer Multiplikation mit einem festen Polynom (Abb. 4.27).

In der abschließenden AddRoundKey-Transformation wird der aus dem geheimen Schlüssel ermittelte Rundenschlüssel mit dem Array durch ein bitweises XOR verknüpft (Abb. 4.28).

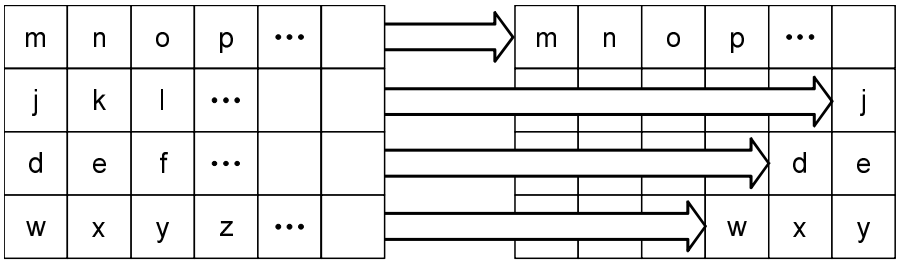

Abb. 4.26: ShiftRow-Transformation

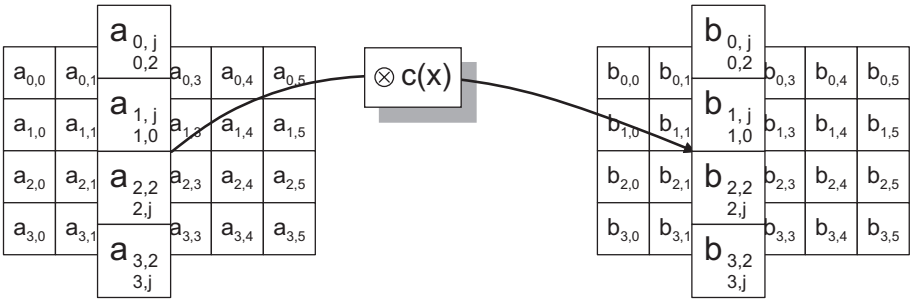

Abb. 4.27: MixColumn-Transformation

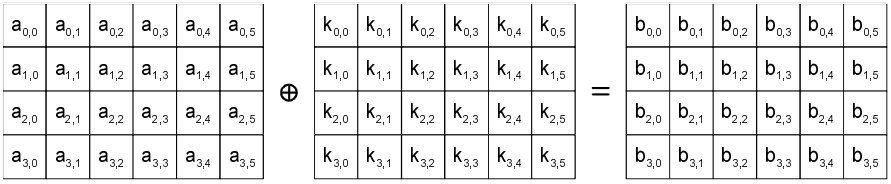

Abb. 4.28: AddRoundKey-Transformation

In der letzten Runde von Rijndael wird die MixColumn-Transformation überschlagen und direkt in die AddRoundKey-Transformation verzweigt.

Die in den einzelnen Runden benutzten Rundenschlüssel werden aus dem originalen Schlüssel durch eine Expansions-Funktion berechnet. Über XOR, zyklische Shifts und einen Tabellen-Lookup werden vor Beginn der Ver- beziehungsweise Entschlüsselung alle Rundenschlüssel berechnet. Dabei wird ein Puffer der Länge (Blocklänge in Bit) * (Anzahl der Runden +1) gefüllt, aus dem die jeweiligen Rundenschlüssel dann entnommen werden. Die ersten Ns Bits (Ns = Schlüssellänge) des Puffers entsprechen dem Schlüssel in unverfälschter Form, alle anderen jeweils Ns Bits entstehen aus den vorherigen Ns Bits durch eine zyklische Permutation und eine Transformation, die der oben angegebenen ByteSub-Transformation ähnelt.

Vor dem Beginn der ersten Runde wird eine initiale AddRoundKey-Transformation durchgeführt, die den Klartext mit dem ersten Rundenschlüssel verknüpft.

Die Entschlüsselung erfolgt analog zur Verschlüsselung, für jede der Transformationen können inverse Transformationen angegeben werden. Jede Runde wird nach folgendem Schema invertiert:

- AddRoundKey,
- InvMixColumn,
- InvShiftRow,
- InvByteSub

Analog zur Verschlüsselung entfällt in der letzten Runde die InvMixColumn-Transformation.

Bewertung der Sicherheit

AES ist der Standard für symmetrische Verschlüsselung in den nächsten Jahren. Seine Schlüssellänge von maximal 256 Bit macht ihn für die nächste Zeit gegen Brute-Force-Attacken unempfindlich. Da der Algorithmus bei dem AES-Auswahlverfahren weltweit gründlich analysiert wurde, ist ein besserer Angriff als Brute-Force nicht zu erwarten. Bei Neuanschaffungen von VPNs sollte die Verfügbarkeit von AES ein Kriterium zum Kauf sein.

4.2.3 Asymmetrische Verschlüsselungs-Verfahren

Asymmetrische Verschlüsselungs-Verfahren sind etwa um den Faktor 1000 bis 10 000 langsamer als symmetrische Verfahren, weshalb sie innerhalb von VPNs nur zur Authentikation und zum Austausch des Schlüssels für die symmetrische Datenverschlüsselung benutzt werden.

4.2.3.1 Diffie-Hellman

Diffie-Hellman war der erste Algorithmus, der auf einer Kombination von öffentlichen und privaten Schlüsseln beruht. Er wurde 1976 von Whitfield Diffie und Martin Hellman entwickelt und beruht auf der Tatsache, dass eine Potenzierung von großen Zahlen wesentlich einfacher ist als die Umkehrfunktion, der Logarithmus. Diffie-Hellman wurde ursprünglich dazu entwickelt, einen Schlüsselaustausch zwischen zwei oder mehr Parteien vorzunehmen, ohne dass dieser Schlüssel von Unbefugten aus der (abgehörten) Kommunikation zwischen den Partnern ermittelt werden kann.

Beschreibung des Verfahrens

Zunächst soll der einfache Fall eines Schlüsselaustauschs zwischen zwei Personen dargestellt werden. Beide einigen sich auf eine große Primzahl n und eine natürliche Zahl g. Diese können über ein unsicheres Netz ausgetauscht werden, ohne

dass eine Gefahr besteht. Jetzt wählen beide Partner je eine geheime Zahl, die hier als x und y bezeichnet werden.

Der Besitzer von x bildet

$X=g^x \bmod n$

und sendet X über die Leitung zu seinem Partner. Der Besitzer von y bildet

$Y=g^y \bmod n$

und sendet Y ebenfalls an den Partner. Ein potentieller Angreifer, der die gesamte Kommunikation abgehört hat, kann aus n, g, X und Y wegen der nicht trivialen Umkehrfunktion zu g^x und g^y die geheimen Zahlen x und y nicht berechnen.

Nach Empfang von X und Y berechnet der eine Partner

$k=Y^x \bmod n$,

der andere

$k'=X^y \bmod n$,

die beide gleich sind:

$k = k' = g^{xy} \bmod n$.

Also ist k bzw. k´ der gesuchte geheime Schlüssel. Analog lässt sich das Verfahren mit mehr als zwei Teilnehmern durchführen, der gesuchte Schlüssel ist beispielsweise bei vier Personen w,x,y und z

$k = g^{wxyz} \bmod n$.

Mit dieser Methode lässt sich aber auch ein übliches Schlüsselpaar aus privatem und öffentlichem Schlüssel generieren. Dabei muss die Primzahl n und die natürliche Zahl g allen potentiellen Teilnehmern bekannt sein. Jeder Benutzer wählt per Zufallsgenerator einen geheimen Schlüssel x. Der dazugehörige öffentlichen Schlüssel wird dann durch

$X = g^x \bmod n$

gebildet. Alle öffentlichen Schlüssel X´ können beispielweise über das Internet ausgetauscht bzw. von einem LDAP-Server bei Bedarf geladen werden. Will ein Teilnehmer einem anderen eine verschlüsselte Nachricht senden, ermittelt er aus seinem geheimen Schlüssel und dem öffentlichen Schlüssel des Gegenübers den zur (symmetrischen) Verschlüsselung benutzten Schlüssel k. Der Empfänger kann k ebenfalls aus seinem geheimen Schlüssel und dem öffentlichen Schlüssel des Senders generieren.

Die bei Diffie-Hellman benutzten mathematischen Operationen sind den üblichen Mikroprozessoren nicht gerade auf den Leib geschrieben, was die nicht sehr große Geschwindigkeit des Verfahrens erklärt.

Bewertung der Sicherheit

Die Sicherheit von Diffie-Hellman hängt primär von der Wahl der Primzahl n und der natürlichen Zahl g ab. Für n (und auch für die geheimen Schlüssel x) gilt »Je größer, desto besser«, denn ein Brute-Force-Angriff muss ja aus

$X=g^x \bmod n$

den Wert von x ermitteln. Die Zahl n sollte mindestens 1024 oder 2048 Bit lang sein, um eine gute Sicherheit zu gewährleisten.

Die Zahl g unterliegt nur der Einschränkung, dass sie primitiv modulo n ist. Ihre Größe trägt nichts zur Sicherheit des Verfahrens bei, so dass bei praktischen Implementierungen oft kleine g im einstelligen Bereich benutzt werden.

4.2.3.2 RSA

Der RSA-Algorithmus wurde nach seinen Entdeckern Ron Rivest, Adi Shamir und Leonhard Adleman benannt. Sie stellten ihn 1978 der Öffentlichkeit vor. RSA kann als das erste Verfahren beeichnet werden, das für Verschlüsselung und digitale Signatur innerhalb einer Public Key Infrastructure (PKI) entworfen wurde, wenngleich dieses Wort damals noch niemand in den Mund nahm. Diffie-Hellman ist zwar älter als RSA, doch wurde das Verfahren zunächst zum Schlüsselaustausch entworfen. Die oben beschriebene Möglichkeit zum Einsatz in einer PKI wurde erst später entwickelt.

Beschreibung des Verfahrens

RSA beruht ähnlich wie Diffie-Hellman auf mathematischen Operationen, deren Umkehrung extrem aufwendig ist. Hier ist es das Produkt zweier großer Primzahlen, das sich schnell bilden lässt. Die Faktorisierung eines solchen Produktes ist hingegen sehr aufwendig.

Um den öffentlichen und den privaten Schlüssel zu berechnen, werden zwei große Primzahlen p und q zufällig ausgewählt und das Produkt pq gebildet. Anschließend können p und q gelöscht werden. Der öffentliche Schlüssel besteht zum einen aus diesem Produkt n=pq, zum anderen aus dem Chiffrierschlüssel e, der wieder mittels Zufallsgenerator bestimmt wird. Es muss ein e gefunden werden, das zu dem Produkt (p-1)(q-1) prim ist, das heißt, es gibt keine gemeinsamen Teiler. Der private Schlüssel d berechnet sich dann zu

$d = e^{-1} \bmod ((p-1)(q-1))$.

Zur Verschlüsselung wird der Klartext in kleine Blöcke b zerlegt und diese werden alle nacheinander mit dem öffentlichen Schlüssel (n und e) auf die verschlüsselten Blöcke v mittels

$v = b^e \bmod n$

abgebildet. Die Entschlüsselung geschieht mit dem privaten Schlüssel d durch

$b = v^d \bmod n.$

Außer einer Verschlüsselung kann mit RSA auch eine digitale Signatur durchgeführt werden, Dazu wird zunächst einmal ein Hashwert über das zu unterzeichnende Dokument gebildet. Dieser Hashwert h wird dann mit dem privaten Schlüssel d verschlüsselt:

$v = h^d \bmod n.$

Der Empfänger bildet den Hashwert aus dem Original-Dokument und vergleicht ihn mit dem durch den öffentlichen Schlüssel (e und n) entschlüsselten Hashwert:

$h = v^e \bmod n.$

Stimmen beide überein, muss das Dokument vom Besitzer des privaten Schlüssels signiert worden sein, der in aller Regel der Absender der Datei sein sollte.

Bewertung der Sicherheit

Das Problem für einen Angreifer ist die Gewinnung des privaten Schlüssels d aus dem ihm bekannten öffentlichen Schlüssel e beziehungsweise n. Kennt er die beiden Primzahlen p und q, ist der Schlüssel geknackt. Das Knacken von RSA reduziert sich somit auf die Primzahlfaktorierung großer Zahlen. Wegen der Größe der bei RSA verwendeten Zahlen (1024 Bit, besser aber 2048 oder 4096 Bit), ist ein Brute-Force-Angriff durch Ausprobieren aller Möglichkeiten

$n = pq$

praktisch ohne Aussicht.

Es existiert jedoch ein Angriff schneller als Brute-Force. Ein schwedisches Forscherteam konnte mit Hilfe des Verfahrens »General Number Field Sieve« eine extrem schnelle Faktorierung von n durchführen und so 512 Bit RSA-Schlüssel knacken. RSA mit 512 Bit darf deshalb keinesfalls mehr zur Anwendung kommen.

Schlüsselqualität:

Die Sicherheit des RSA-Verfahrens beruht auf der Schwierigkeit, eine große Zahl in ihre Primfaktoren zu zerlegen. In der Literatur findet man mehrere Ansätze, dieses Problem zu lösen.

Beispielsweise gibt es Methoden, die auf der Kenntnis der (teilweisen) Faktorisierung von p-1 oder p+1 (gleiches gilt für q-1 und q+1) beruhen. Zahlen, die neben ihrer Primzahleigenschaft noch mehrere die Sicherheit erhöhende Merkmale wie große Primteiler in p-1 und p+1 haben und die somit diese Methoden erschweren, werden daher als »starke« Primzahlen bezeichnet.

up ...

... up ... update

Nutzen Sie den UPDATE-SERVICE des mitp-Teams bei vmi-Buch.

Registrieren Sie sich JETZT!

Unsere Bücher sind mit großer Sorgfalt erstellt. Wir sind stets darauf bedacht, Sie mit den aktuellsten Inhalten zu versorgen, weil wir wissen, dass Sie gerade darauf großen Wert legen. Unsere Bücher geben den top-aktuellen Wissens- und Praxisstand wieder.

Um Sie auch über das vorliegende Buch hinaus regelmäßig über die relevanten Entwicklungen am IT-Markt zu informieren, haben wir einen besonderen Leser-Service eingeführt.

Lassen Sie sich professionell, zuverlässig und fundiert auf den neuesten Stand bringen.
Registrieren Sie sich jetzt auf www.mitp.de oder **www.vmi-buch.de** und Sie erhalten zukünftig einen E-Mail-Newsletter mit Hinweisen auf Aktivitäten des Verlages wie zum Beispiel unsere aktuellen, kostenlosen Downloads.

Ihr Team von mitp

An die zu suchende Zahl p (bzw. q) werden folgende Forderungen gestellt /Gord84/:

- p ist eine große Zahl
- p ist eine Primzahl
- p wurde zufällig ausgewählt
- p hat eine vorher festgelegte Länge
- p-1 hat einen großen Primteiler r
- p+1 hat einen großen Primteiler s
- r-1 hat einen großen Primteiler
- s-1 hat einen großen Primteiler.

4.2.3.3 ElGamal und DSA

Der von Taher ElGamal 1993 entwickelte und nach ihm benannte Algorithmus ist eine Variante von Diffie-Hellman. Er beruht auf dem gleichen Prinzip, dass Potenzen leicht, die Umkehrfunktion Logarithmus dagegen schwer zu berechnen ist. Im Unterschied zu Diffie-Hellman wurde ElGamal direkt für den Einsatz in PKIs entwickelt. Das klassische ElGamal dient der digitalen Signatur, es gibt allerdings auch eine modifizierte Variante, mit der Verschlüsselung möglich ist. In Europa wird ElGamal nur selten eingesetzt, aber in den USA hat es durch seine Implementierung im staatlichen Signaturverfahren DSA (Digital Signatur Algorithm) eine große Verbreitung.

Beschreibung des Verfahrens

Jeder Benutzer bestimmt zunächst einen privaten und einen öffentlichen Schlüssel. Dazu werden per Zufallsgenerator eine große Primzahl p und zwei Zahlen g und x bestimmt, wobei gelten muss

$g < p$ und $x < p$.

Der private Schlüssel ist x.

Mit

$y = g^x \bmod p$

wird der öffentliche Schlüssel bestimmt, der aus dem Zahlentripel g, p und y besteht. Um ein Dokument (oder vielmehr dessen Hash) h zu signieren, wird zunächst eine weitere Zufallszahl k bestimmt, die keinen gemeinsamen Teiler mit (p-1) hat.

Der erste Teil der Signatur besteht aus der Zahl a mit

$a = g^k \bmod p$,

zur Berechnung des zweiten Teils b muss folgende Gleichung nach b ausgelöst werden:

$h = (xa + kb) \bmod (p-1)$.

Gilt dann

$y^a a^b \bmod p = g^h \bmod p$,

so ist die Unterschrift authentisch.

Bewertung der Sicherheit

Ein Brute-Force-Angriff auf ElGamal muss aus der Gleichung

$y = g^x \bmod p$

den geheimen Schlüssel x bestimmen. Werden große Zahlen benutzt (1024 oder 2048 Bit), sind diese Angriffe praktisch aussichtslos. Hier ähnelt ElGamal seinem Vorbild Diffie-Hellman stark. Es existiert aber noch ein anderer möglicher Angriff gegen das Verfahren, der weit weniger aufwändig ist. Wird die Zufallszahl k von einem schlechten Zufallsgenerator bestimmt, könnte es dem Angreifer gelingen, zwei Nachrichten-Hashwerte h und h´ mitzuschneiden, von denen er weiß, dass sie mit demselben k signiert wurden. Nachrichten mit demselben k sind daran zu erkennen, dass sie dieselbe Teilunterschrift

$a = g^k \bmod p$

tragen. Vom Gleichungssystem

$h = (xa + kb) \bmod (p-1)$

$h' = (xa + kb') \bmod (p-1)$

sind alle Größen außer x und k bekannt. Der gesuchte private Schlüssel x lässt sich daraus ohne großen Aufwand berechnen. Da eine Wiederholung von k bei den heutigen Zufallsgeneratoren durchaus im Bereich des Möglichen liegt, ist es rätselhaft, weshalb dieser Algorithmus im staatlichen Bereich in den USA vorgeschrieben ist. Vielleicht wollten sich die Geheimdienste eine zumindest theoretische Hintertür offen halten.

4.2.4 Hash-Verfahren

Ein Hashwert dient innerhalb einer VPN-Infrastruktur als Vorbereitung der digitalen Signatur einer Menge von Daten. Er komprimiert die gegebene Datenmenge auf eine feste, kleine Länge und kann als eine Art Prüfsumme bezeichnet werden. Dabei gelten aber besondere Anforderungen an den Hash-Algorithmus:

- Die Hash-Funktion ist eine Einbahnstraße, das heißt, es ist leicht, aus einer gegebenen Datenstruktur deren Hashwert zu berechnen. Umgekehrt ist es praktisch unmöglich, aus einem gegebenen Hashwert eine Datenstruktur zu berechnen, die eben diesen Hashwert hat.
- Es ist extrem schwierig, zu einer gegebenen Datenmenge eine andere Datenmenge zu berechnen, die denselben Hashwert besitzt.
- Eine kleine Änderung in der Datenmenge hat große Auswirkungen auf den Hash.

Mit diesen Anforderungen sollen Manipulationen an den Originaldaten beziehungsweise am Hashwert sofort sichtbar gemacht werden, so dass der Hashwert zur Konsistenzüberprüfung der Daten dienen kann. Wird der Hashwert zusätzlich noch signiert, kann der Absender der Daten nachvollziehbar überprüft werden. Alternativ kann der Hashwert mit einem geheimen Schlüssel manipuliert und dem Empfänger zugestellt werden. Besitzt dieser den Schlüssel ebenfalls, kann er den Hashwert kontrollieren. Diese Kombination aus Hashwert und geheimem Schlüssel wird auch als Message Authentication Code (MAC) bezeichnet.

4.2.4.1 Message Digest 4 (MD4)

Ron Rivest entwickelte Anfang der neunziger Jahre das MD4-Verfahren. Heute gilt es nicht mehr als besonders sicher, doch da Microsoft es in seiner Windows-Umgebung zur Authentikation, zum Verschlüsseln der Passworte in der SAM-Datenbank und in den Microsoft-eigenen VPNs benutzt, darf es an dieser Stelle nicht fehlen. MD4 erzeugt aus Dokumenten beliebiger Länge einen Hashwert von 128 Bit.

Beschreibung des Verfahrens

MD4 ist ein Algorithmus, der im Wesentlichen aus nichtlinearen Funktionen besteht. Diese enthalten logische Verknüpfungen der booleschen Funktionen AND, OR, NOT und XOR. MD4 zerlegt den Text in Blöcke zu 512 Bit, der letzte Block wird gegebenenfalls aufgefüllt. Vier Variablen A,B,C und D (je 32 Bit) werden mit festen Werten initialisiert und in drei Runden über nichtlineare Funktionen mit den Textblöcken verknüpft. A,B,C und D ergeben sich in den Runden 2 und 3 aus den Ergebnissen der vorhergehenden Runde. Abschließend werden die Ergebnisse der Runde 3 zu einem 128-Bit-Hashwert zusammengestellt.

Der Algorithmus im Detail:

Der Text wird zunächst auf eine Länge von n * 512 – 64 gebracht, wobei das Auffüllen (Padding) mit einer 1 und nachfolgenden 0 vorgenommen wird. Die letzten 64 Bit enthalten dann die Länge des Textes beziehungsweise nur die niederwertigen 64 Bit, falls er länger als 2^{64} ist.

Die vier Zahlen A,B,C und D werden mit Konstanten initialisiert:

A = 01 23 45 67

B = 89 ab cd ef

C = fe dc ba 98

D = 76 54 32 10

In jeder der drei Runden werden die Werte der aktuellen A,B,C und D temporären Variablen AA, BB, CC und DD zugewiesen. Anschließend werden diese Größen über eine der drei Funktionen jeweils 16-mal mit dem Text verknüpft und geshiftet:

Runde 1: F(x,y,z) = (x AND y) OR (NOT(x) AND z)

Runde 2: G(x,y,z) = (x AND y) OR (x AND z) OR (y AND z)

Runde 3: H(x,y,z) = x XOR y XOR z

Dabei ändern sich die AA, BB, CC und DD. Nach der letzten Runde werden diese Werte zu den alten Konstanten A, B, C, D addiert. Die Nebeneinanderstellung der neuen Werte von A, B, C und D ergibt dann den 128-Bit-Hashwert (Abb. 4.29).

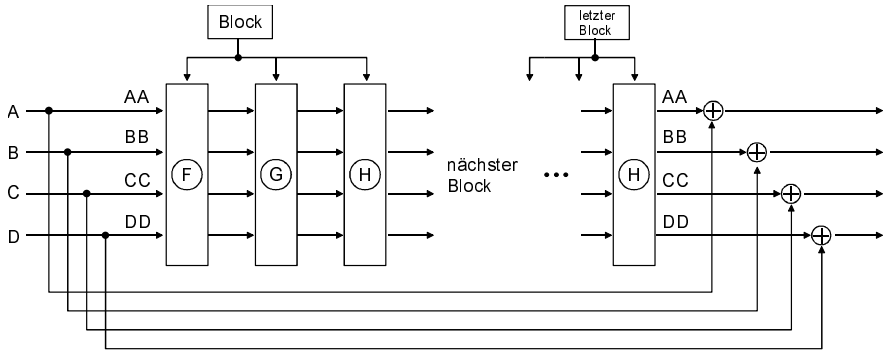

Abb. 4.29: Hashwert-Bildung mittels MD4

Bewertung der Sicherheit

Obwohl MD4 bis heute nicht gebrochen werden konnte, sind Angriffe gegen die ersten beiden Runden möglich. Die Sicherheit des Verfahrens hängt deshalb »am seidenen Faden«, an der dritten Runde.

MD4 sollte aus diesem Grund in neu konzipierten Systemen nicht mehr eingesetzt werden.

4.2.4.2 Message Digest 5 (MD5)

MD5 ist ein verbesserter Nachfolger von MD4 und wurde Anfang der neunziger Jahre ebenfalls von Ron Rivest entwickelt. Er liefert ebenfalls 128-Bit-Hashwerte und ähnelt MD4 von seinem Aufbau sehr, es wurde aber eine vierte Runde hinzugefügt. Außerdem ändern sich A, B, C, und D schon in jeder Runde.

Beschreibung des Verfahrens

Das Padding der zu hashenden Datenstruktur geschieht identisch zu MD4, also durch Auffüllen auf ein Vielfaches von 512 Bit mit einer Eins und Nullen sowie Berechnung der 64 Bit-Längenangabe.

Die Variablen A, B, C und D werden identisch zu MD4 initialisiert, die Funktionen der einzelnen Runden werden um eine vierte erweitert. Dabei wurde G modifiziert, um weniger symmetrisch als bei MD4 zu sein.

Runde 1: F(x,y,z) = (x AND y) OR (NOT(x) AND z)

Runde 2: G(x,y,z) = (x AND y) OR (y AND NOT(z))

Runde 3: H(x,y,z) = x XOR y XOR z

Runde 4: I(x,y,z) = y XOR (x OR (NOT(z))

Die entscheidende Verbesserung liegt aber in der Behandlung der A,B,C und D. Sie werden nach jeder Runde um die AA, BB, CC und DD vergrößert und anschließend den AA, BB, CC und DD für die nächste Runde zugewiesen. Der Hash ergibt sich schließlich durch die A, B, C und D der letzten Runde (Abb. 4.30).

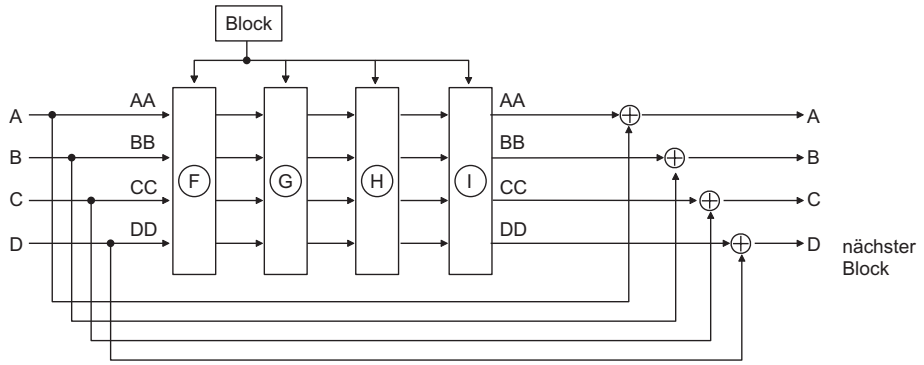

Abb. 4.30: Hashwert-Bildung mittels MD5

Bewertung der Sicherheit

MD5 ist in jedem Fall sicherer als MD4, wenngleich Kryptologen auch bei MD5 Angriffe gegen einige Runden durchführen können. Die bisher veröffentlichten Angriffe beeinträchtigen die Sicherheit von MD5 bisher nicht, doch raten einige Wissenschaftler von der Verwendung von MD5 ab.

4.2.4.3 Secure Hash Algorithm (SHA)

Unter dem Namen SHA sind verschiedene Verfahren zur Hashbildung bekannt, die ab Anfang der neunziger Jahre von den beiden US-amerikanischen Institutionen NSA und NIST entwickelt wurden. Ziel war es, eine Hashfunktion für den Signatur-Standard DSA bereitzustellen. Zusätzlich enthält DSA noch die Signatur des Hashwerts mittels ElGamal.

Der erste dieser Algorithmen, SHA-1 genannt, erzeugt einen Hashwert von 160 Bit Länge. Urmutter war auch hier der MD4-Algorithmus, wobei die Gemeinsamkeiten geringer sind als zwischen MD4 und MD5. Die Nachfolger von SHA-1 erzeugen aufgrund kleiner Modifikationen am Verfahren Hashwerte der Längen 256 Bit,

384 Bit und 512 Bit. Diese Algorithmen werden dementsprechend auch als SHA-256, SHA-384 und SHA-512 bezeichnet. Wird nur von SHA gesprochen, so ist in der Regel der alte SHA-1 gemeint.

Beschreibung des Verfahrens (SHA–1)

Das Padding der Eingangsdaten geschieht absolut identisch zu MD4 und MD5, das heißt durch Auffüllen auf ein ganzzahliges Vielfaches von 512 Bit inklusive 64 Bit Längenangabe. Wegen der Hashlänge von 160 Bit benutzt SHA–1 fünf Variablen zu je 32 Bit, die wie folgt initialisiert werden:

A = 67 45 23 01

B= ef cd ab 89

C = 98 ba dc fe

D = 10 32 54 76

E = c3 d2 e1 f0

Ein entscheidender Unterschied zwischen SHA–1 und MD4/MD5 ist eine Expansion der Nutzdaten, bei der bei jedem 512-Bit-Block nach einer Zerlegung in 16 * 32-Bit durch XOR und Shift-Operationen zusätzlich 72 weitere 32-Bit-Blöcke berechnet und in den Algorithmus eingebracht werden. Diese Funktion, die aus den 16 Wörtern insgesamt 80 Wörter macht, wird von Kryptologen als starke Verbesserung der Qualität des Verfahrens bezeichnet.

Es werden wieder 4 Runden durchlaufen, in jeder Runde aber 20 Operationen. In der ersten Runde werden dabei die ersten 20 32-Bit-Blöcke benutzt, in der zweiten Runde die Blöcke 21 bis 30 usw. In jeder Runde werden die Konstanten A, B, C, D und E wieder temporären Größen AA, BB, CC, DD und EE zugewiesen, die mit den Nutzdaten durch Shift-Operationen und nichtlineare Funktionen neue AA etc. berechnen. Diese werden dann wie bei MD5 auf die Größen A, B, C, D und E addiert. Die nichtlinearen Funktionen haben die folgende Form:

Runde 1: F(x,y,z) = (x AND y) OR (NOT(x) AND z) (identisch zu MD4 und MD5)

Runde 2: G(x,y,z) = x XOR y XOR z

Runde 3: H(x,y,z) = (x AND y) OR (x AND z) OR (y AND z)

Runde 4: I(x,y,z) = x XOR y XOR z

Die zuletzt berechneten Werte A, B, C, D und E werden analog zu MD4/MD5 zum 160 Bit-Hashwert kombiniert (Abb. 4.31).

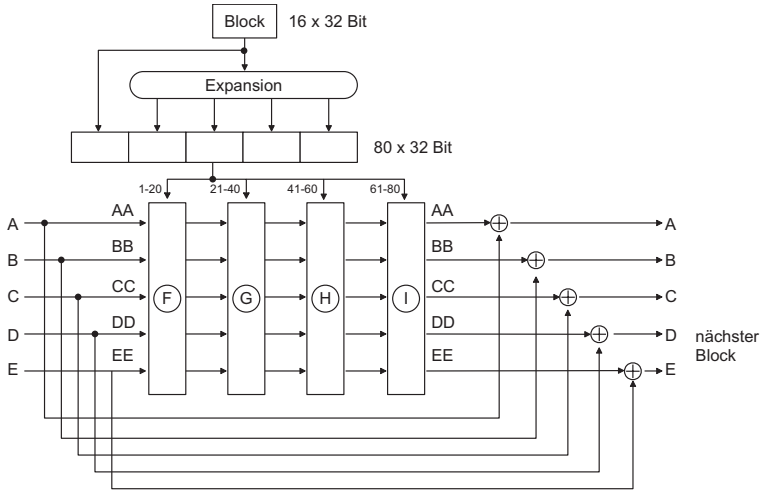

Abb. 4.31: Hashwert-Bildung mittels SHA-1

Bewertung der Sicherheit

Wegen des längeren Hashwerts sind Brute-Force-Angriffe vor allem gegen die SHA-Varianten mit 384 und 512 Bit ungleich schwieriger als gegen MD4/MD5. Angriffe gegen SHA, die bessere Möglichkeiten bieten als Brute-Force, sind bisher nicht bekannt.

Bis auf die alte 160-Bit-Variante SHA-1 wird SHA deshalb zusammen mit dem symmetrischen AES die Kryptosysteme der nächsten Jahre dominieren und auch in vielen VPN-Lösungen anzutreffen sein.

4.2.4.4 HMAC

Als Beispiel für ein Message Authentication Code-Verfahren (MAC) soll der 1997 bei IBM entwickelte Algorithmus HMAC beschrieben werden. Wie andere MAC-Verfahren kombiniert HMAC eine Hash-Funktionen mit symmetrischer Verschlüsselung. Da je nach Implementierung des VPN jedes einzelne Netzwerkpakete mittels MAC abgesichert werden muss, ist die Geschwindigkeit des Verfahrens eine wichtige Kenngröße. Eine reine Hintereinanderschaltung von Hash und Verschlüsselung wäre aus Gründen der Performance ungünstig, so dass die Entwickler von HMAC folgende Randbedingungen berücksichtigen mussten:

- Das Verfahren sollte mit möglichst vielen Hash-Funktionen zusammenarbeiten, ohne dass diese modifiziert werden mussten.
- Die Geschwindigkeit der Hash-Algorithmen sollte nur unwesentlich verlangsamt werden.
- Die Sicherheit des Hash-Verfahrens durfte durch die Manipulationen mit dem geheimen Schlüssel nicht verringert werden.

Beschreibung des Verfahrens

Ausgangspunkt von HMAC ist eine Hash-Funktion H(Text), etwa MD5 oder SHA–1. Das zu bearbeitende Dokument (Text) wird mit dem geheimen Schlüssel K und zwei internen Datenstrukturen ipad und opad nach folgender Gleichung kombiniert:

HMAC = H(K XOR opad, H(K XOR ipad, text)) mit

ipad = 0x36, 0x36, 0x36 ...

opad = 0x5C, 0x5C, 0x5C ...

Die Felder ipad und opad haben eine Länge, die der Blockgröße B des eingesetzten Hash-Algorithmus entspricht (64 Bytes bei MD5 und SHA–1). Der Schlüssel K wird durch das Anhängen von Nullen ebenfalls auf diese Länge B gebracht.

Die obige Gleichung bedeutet im Einzelnen:

- Verknüpfe den auf die Länge B gebrachten geheimen Schlüssel K mittels XOR mit der ipad-Datenstruktur.
- Stelle das Ergebnis dieser Operation vor das zu verschlüsselnde Dokument und schicke diesen so um B vergrößerten Text durch die Hash-Funktion H.
- Das Ergebnis dieser Hash-Operation hat die Länge L (16 Bytes bei MD5, 20 Bytes bei SHA–1).
- Verknüpfe den auf die Länge B gebrachten geheimen Schlüssel K mittels XOR mit der opad-Datenstruktur.
- Stelle das Ergebnis dieser Operation (Länge B) vor das Hash-Ergebnis (Länge L) und schicke diese Bytestrom der Länge L+B durch die Hashfunktion H.
- Der HMAC-Hash wird zusammen mit dem verschlüsselten Dokument an den Empfänger geschickt.

Abbildung 4.32 verdeutlicht diesen Sachverhalt.

Die Geschwindigkeit von HMAC ist bei größeren Datenstrukturen (Text) nur unwesentlich kleiner als die der verwendeten Hash-Funktion, obwohl diese zweimal aufgerufen wird. Beim ersten Hash wird der Text um das Ergebnis der XOR-Operation (Länge B) vergrößert, der zweite Hash arbeitet mit einer Länge L+B und beansprucht nur wenig Ressourcen. Soll bei einem VPN allerdings jedes Netzwerkpaket mittels HMAC abgesichert werden, entstehen relativ zu den kleinen Paketlängen große zusätzliche Aufwendungen.

Der Empfänger des (verschlüsselten) Dokuments kann den HMAC-Hash bilden, sofern er das Klartext-Dokument zurück gewinnen kann und außerdem den geheimen Schlüssel für die HMAC-Operation besitzt. Stimmt der Wert mit der mitgeschickten Größe des HMAC überein, ist das Dokument unverfälscht übertragen worden und stammt tatsächlich von einem Besitzer des geheimen HMAC-Schlüssels.

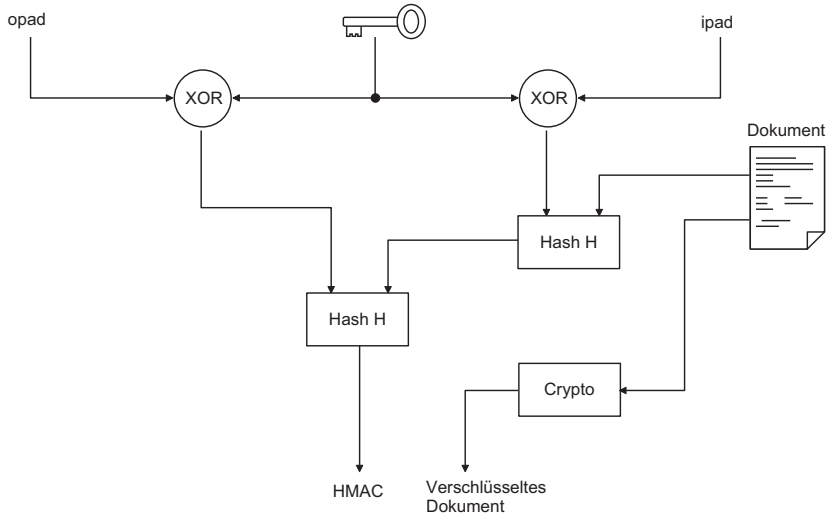

Abb. 4.32: HMAC

Bewertung der Sicherheit des Verfahrens

Der Einsatz von HMAC ist von der Sicherheit her identisch zu dem jeweils eingesetzten Hash-Verfahren.

4.3 Infrastruktur von Zertifizierungs-Systemen

Die zur Absicherung von öffentlichen Schlüsseln eingesetzten Zertifizierungs-Systeme bedürfen einer Infrastruktur, mit der in einer für den Benutzer nachvollziehbaren Weise Schlüssel und Zertifikate verwaltet und kontrolliert werden können. Eine solche Infrastruktur, die häufig als PKI (Public Key Infrastructure) bezeichnet wird, muss folgende Aufgaben erfüllen:

- Vergabe von eindeutigen elektronischen Identifikationen
- Bereitstellung und Verteilung von zertifizierten Schlüsseln
- eindeutige Zuordnung von zertifizierten Schlüsseln zu Personen oder Firmen
- Überprüfung von zertifizierten Schlüsseln
- Verwaltung von ungültigen Zertifikaten, die etwa durch Zeitablauf oder Diebstahl entstehen können

In der Praxis konnten sich eine Reihe von Verfahren etablieren, mit denen eine PKI aufgebaut werden kann. Diese unterscheiden sich vor allem im Zugriff auf die Zertifikate und in der Struktur der Vertrauensverhältnisse der beteiligten Parteien. Auf lange Sicht wird sich die im Standard X.509 festgelegte PKI durchsetzen, doch benötigt dieses Verfahren einen eigenen Verzeichnisdienst zum Zugriff auf Schlüssel und Zertifikate.

Unter den Verzeichnisdiensten, deren Aufbau und Wirkungsweise im Standard X.500 festgelegt wurden, hat sich eine etwas abgespeckte Variante mit dem Protokoll LDAP (Lightweight Directory Access Protocol) im VPN-Bereich durchgesetzt.

4.3.1 X.509-Zertifikate

Am Prozess der Ausgabe, Verwaltung und Vernichtung von Schlüsseln und Zertifikaten sind unter X.509 mehrere Parteien beteiligt.

Die Zertifizierungs-Instanz – auch Certification Authority (CA) genannt – erzeugt Zertifikate und Schlüsselpaare beziehungsweise zertifiziert bestehende (öffentliche) Schlüssel. Sie ist verantwortlich für die eindeutige Zuordnung zwischen Schlüssel, Zertifikat und dem Besitzer des Schlüssels. Der öffentliche Schlüssel ist dabei Bestandteil der Datenstruktur des Zertifikats. Die CA versieht die Zertifikate mit einer digitalen Signatur und einem Verfallsdatum, ab dem diese für ungültig erklärt werden. Falls ein Zertifikat vor diesem Datum aus dem Verkehr gezogen werden sollen, wird die Information über den Ablauf ebenfalls von der CA über einen Verzeichnisdienst bekannt gegeben. Wegen des möglichen großen Andrangs bei den CAs können diese einen Teil ihrer Aufgaben, nämlich die Kontrolle der Identität ihrer Kunden, an untergeordnete Registration Authorities (RAs) delegieren. CAs haben die Möglichkeit, sich ihrerseits über übergeordnete CAs abzusichern oder aber selbst ihre eigenen Zertifikate zu unterschreiben. CAs mit selbst unterschriebenen Zertifikaten werden auch als Root-CAs bezeichnet. Gegenüber Root-CAs kann ein Benutzer nur sein grundsätzliches Vertrauen aussprechen oder verweigern, eine Kontrolle der Integrität einer Root-CA ist nicht möglich. Abbildung 4.33 zeigt eine mögliche PKI.

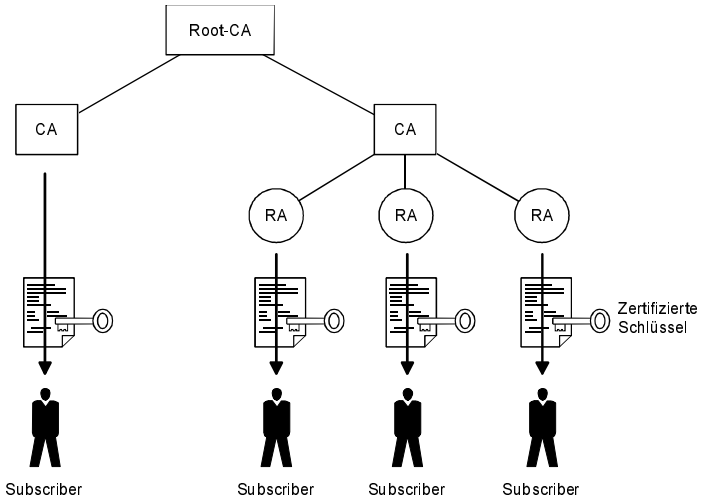

Abb. 4.33: PKI nach X.509

Der Kunde der CA – auch Subscriber genannt – erhält von der CA das Zertifikat, mit dem sein öffentlicher Schlüssel beglaubigt wird. Er muss seine Identität gegenüber der CA bzw. RA nachweisen, wobei die Qualität dieses Nachweises höchst unterschiedlich sein kann. In den meisten Fällen muss er eine Kopie seines Ausweises zur Verfügung stellen, bei so genannten Demo-Zertifikaten reicht allerdings auch die Angabe seiner E-Mail-Adresse aus. Je nach der Qualität seines Identitäts-Nachweises erhält er dann ein Zertifikat einer bestimmten Sicherheits-Kategorie.

Abbildung 4.34 zeigt als Beispiel ein Zertifikat, das von der kommerziellen CA GlobalSign ausgestellt wurde und zur Absicherung von Web- und E-Mail-Zugriffen eingesetzt werden kann. Die Sicherheitskasse 2 bedeutet bei GlobalSign, dass eine (nicht beglaubigte) Kopie des Personalausweises vorgelegt werden musste.

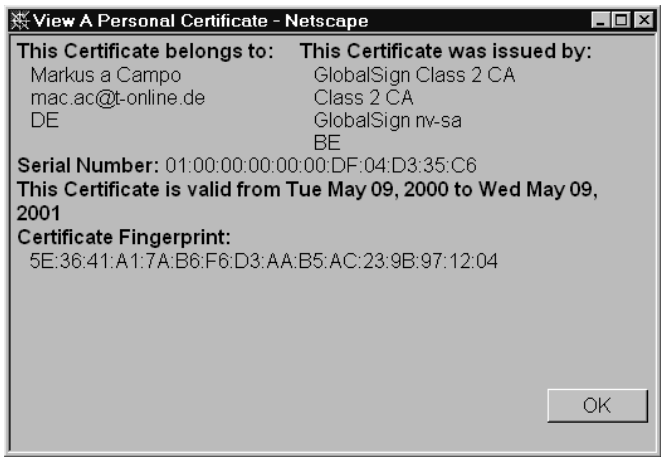

Abb. 4.34: Zertifikat nach X.509

Der Subscriber ist für die Geheimhaltung seines privaten Schlüssels selbst verantwortlich. Dieser wird in den meisten Fällen auf seinem PC abgelegt und durch eine Passphrase (eine Folge von einem oder mehreren Passworten) vor dem Zugriff Unberechtigter geschützt.

Der User schließlich ist die Instanz, die mit der Hilfe des zertifizierten öffentlichen Schlüssels dem Subscriber eine verschlüsselte Nachricht zukommen lassen will oder von diesem eine digital signierte Nachricht erhält. Er muss sich auf die Qualität des Zertifikats und dessen Gültigkeit verlassen können. In die meisten Browser ist eine Anzahl von CAs mit deren Root-CAs fest eingebaut, deren Zertifikaten dann vertraut werden kann (Abb. 4.32). Die Abbildungen 4.36 bis 4.38 zeigen die Vertrauenskette vom Subscriber-Zertifikat der Abbildung 4.34 bis zu einer Root-CA mit selbst unterschriebenem Zertifikat.

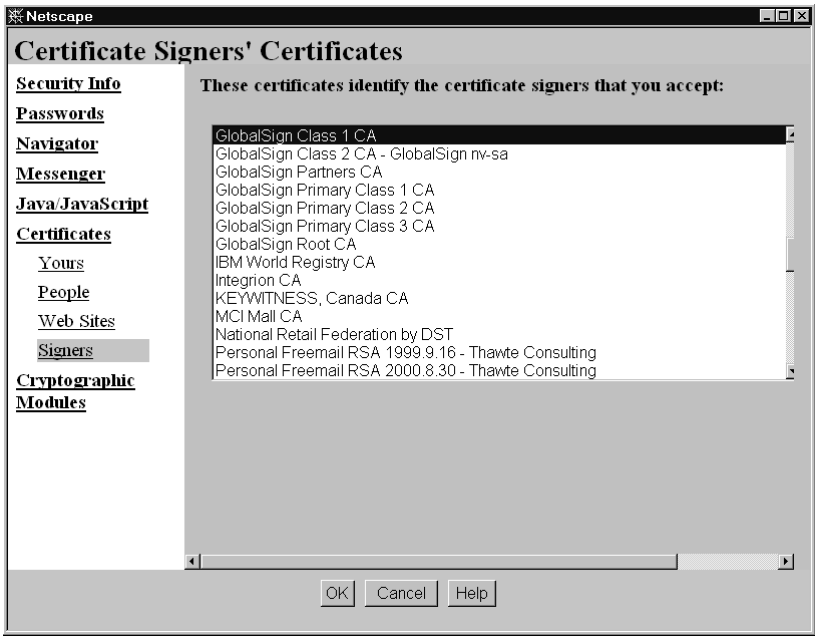

Abb. 4.35: CAs im Netscape-Browser

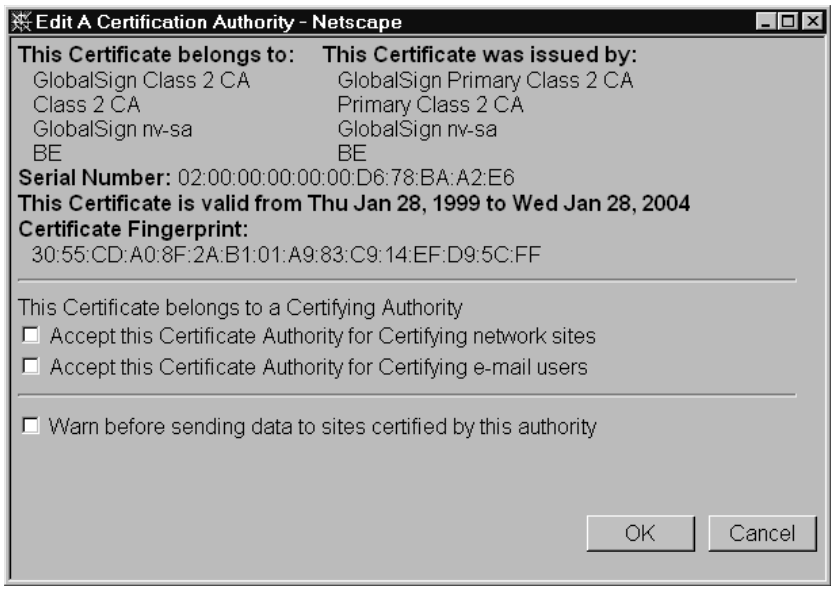

Abb. 4.36: Zertifikat von GlobalSign

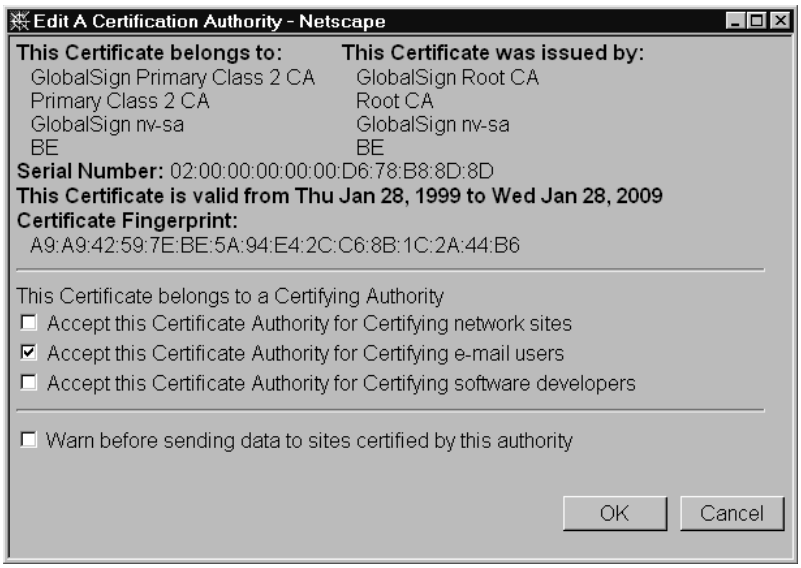

Abb. 4.37: Zertifikat von GlobalSign

Abb. 4.38: Root-Zertifikat von GlobalSign

Aufbau eines X.509-Zertifikats

X.509-Zertifikate bestehen aus drei Teilen, die in Tabelle 4.2 dargestellt werden.

Datenstruktur	Bedeutung
tbsCertificate	Zertifikat-Rumpf
signatureAlgorithm	Algorithmen, mit denen der Hash des Zertifikats gebildet und signiert wurde; mögliche Hash-Funktionen sind MD2 (veraltet!), MD5 oder SHA; signiert wird mit DSA oder RSA
signatureValue	Digitale Signatur der CA

Tabelle 4.2: Aufbau eines X.509-Zertifikats

Der Zertifikat-Rumpf hat die in Tabelle 4.3 dargestellte Struktur.

Datenstruktur	Bedeutung
version	X.509-Version (0, 1 oder 2), 2 bedeutet die aktuelle Version X.509v3
serialNumber	Eindeutige Seriennummer des Zertifikats
signature	Algorithmen, mit denen der Hashwert des Zertifikats gebildet und signiert wurde; identisch zu signatureAlgorithm in Tab. 4.1
issuer	Eindeutiger Name der Institution, die das Zertifikat ausgestellt hat
validity	Gültigkeitsbereich des Zertifikats (von / bis)
subject	Eindeutiger Name des Subscribers (Inhaber des zugeordneten Schlüsselpaars)
subjectPublicKeyInfo	Algorithmus (RSA, DSA oder Diffie-Hellman) und Wert des öffentlichen Schlüssels
issuerUniqueID	optional: eindeutige Identifikation der Institution, die das Zertifikat ausgestellt hat; wird nur benutzt, wenn der im Feld issuer angegebene Name in einem neuen Zusammenhang wieder verwendet werden soll
subjectUniqueID	optional: eindeutige Identifikation des Subscribers; wird nur benutzt, wenn der im Feld subject angegebene Name in einem neuen Zusammenhang wieder verwendet werden soll
extensions	optionale Erweiterungen des Zertifikats

Tabelle 4.3: Zertifikat-Rumpf nach X.509

Mit den im letzten Feld angegebenen optionalen Erweiterungen kann ein X.509-Zertifikat leicht an vorgegebene PKIs adaptiert werden.

Falls die CA, die das Zertifikat ausgestellt hat, sich über eine in der Hierarchie höher stehende CA legitimiert, müssen zwei Zertifikate mit den dazugehörigen Schlüsseln an den User übertragen werden usw.

4.3.2 Pretty Good Privacy (PGP)

Eins der ältesten Verfahren zur Absicherung mittels öffentlicher Schlüssel ist PGP. Der Entwickler von PGP – Phil Zimmermann – begann bereits Mitte der achtziger Jahre mit seiner Arbeit an dem Programm. Die erste lauffähige Version wurde 1991 fertiggestellt. Da PGP frei verfügbar ist, hat es eine große Verbreitung erfahren.

PGP nutzt zur Erzeugung von Schlüsseln bzw. deren Austausch die Verfahren Diffie-Hellman und RSA. Die eigentlichen Daten können wahlweise mittels AES, IDEA, Triple-DES oder CAST (in diesem Buch nicht besprochen) verschlüsselt werden. Der für die digitale Signatur erforderliche Hashwert wird nach SHA-1 oder MD5 gebildet.

Im Unterschied zu X.509 verfügt das klassische PGP nicht über eine streng hierarchische Struktur von Vertrauensverhältnissen. Statt dessen wird ein so genanntes »Web of Trust« (Abb. 4.39) installiert, bei dem sich die Teilnehmer gegenseitig vertrauen. Jeder Benutzer hat auf seinem Rechner einen Schlüsselbund installiert, dessen Elemente die öffentlichen Schlüssel seiner Partner sowie Angaben über die Gültigkeit der Schlüssel und die Vertauenswürdigkeit seiner Besitzer sind (Abb. 4.40). Die Elemente des Schlüsselbunds (Abb. 4.38) entsprechen von ihrer Funktion her den Zertifikaten aus X.509 und können per E-Mail oder über sogenannte Key-Server verteilt werden. Die kommerzielle Implementierung von PGP lässt dem Administrator die Wahl zwischen den PGP-Zertifikaten und einer X.509-konformen Struktur mit hierarchischen Vertrauensverhältnissen.

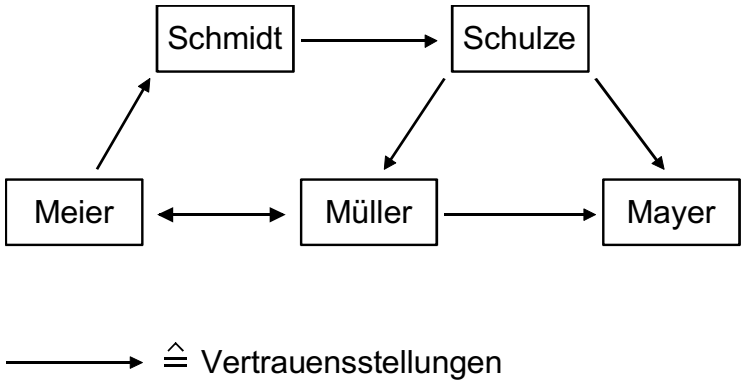

Abb. 4.39: Web of Trust

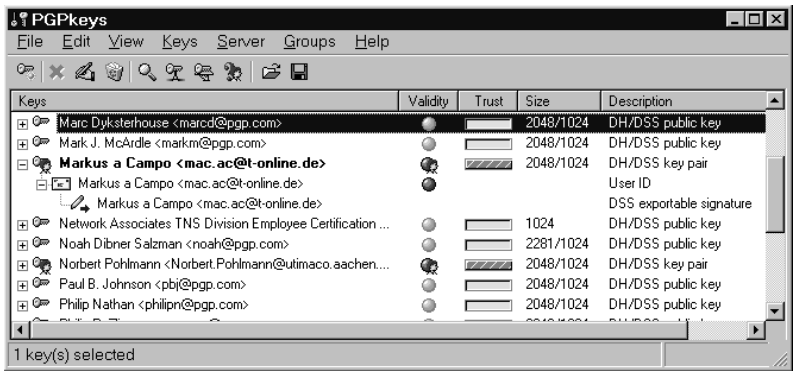

Abb. 4.40: PGP-Schlüsselbund

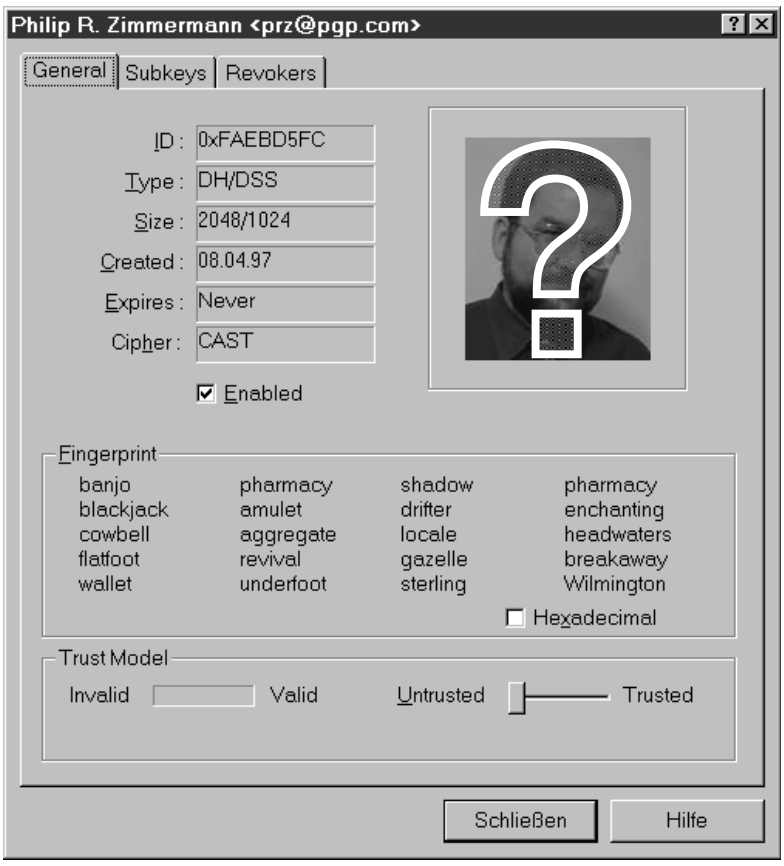

Abb. 4.41: PGP-Zertifikat

4.3.3 Verzeichnisdienste und das LDAP-Protokoll

Ein Verzeichnisdienst ist ein über Netzwerk zugänglicher Server, der über die Angabe von Namen und Attributen nach bestimmten Eigenschaften eines zugeordneten Objekts sucht. Dabei ist die Art, in der die Objekte in das Verzeichnis eingetragen werden können, in weiten Grenzen frei definierbar. In den Standards X.400 und X.500 wurde ein allgemeiner Verzeichnisdienst definiert, der sich für die Praxis im Internet allerdings als zu kompliziert erwiesen hat. Eine vereinfachte Version des Zugriff auf einen Verzeichnis-Servers wird in der LDAP-Spezifikation (Lightweight Directory Access Protocol) beschrieben. Abbildung 4.42 zeigt ein Beispiel für eine allgemein gehaltene Verzeichnis-Struktur, die mittels LDAP im Netzwerk publiziert werden könnte.

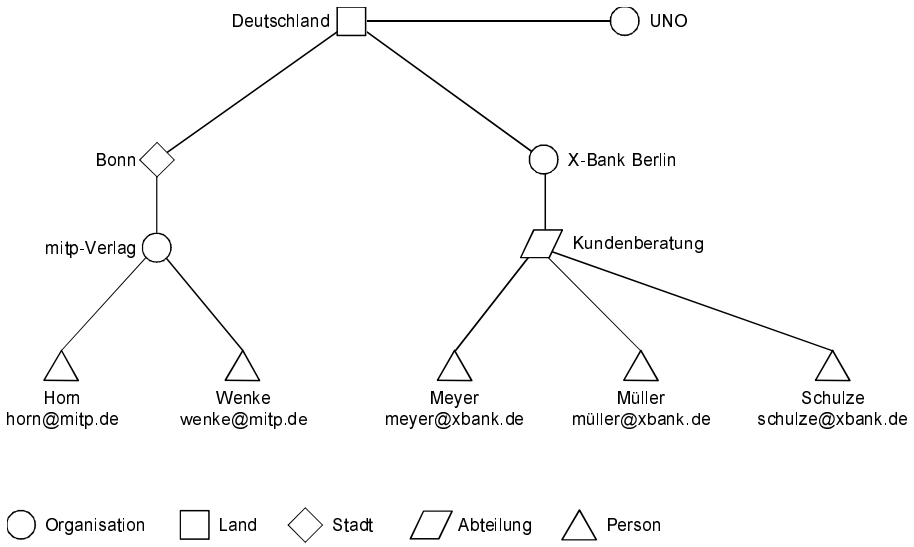

Abb. 4.42: Verzeichnis-Struktur

Die Verwaltung von zertifizierten Schlüsseln ist eine typische Anwendung für einen Verzeichnisdienst. Deshalb greifen Zertifizierungs-Systeme gerne auf diese Infrastruktur zurück. Die Zertifizierung nach dem Standard X.509 ist sogar von ihren Datenstrukturen her auf die Zusammenarbeit mit Verzeichnisdiensten abgestimmt.

Zur Unterscheidung der am Prozess der Zertifizierung und Überprüfung von Schlüsseln beteiligten Instanzen wird der so genannte Distinguished Name (DN) als Unterscheidungsmerkmal zwischen Objekten benutzt.

Diese Datenstruktur besteht ihrerseits aus verschiedenen Komponenten, die beispielsweise die ausstellende Instanz beschreiben (Tab. 4.4):

Schlüsselwort	Kürzel	Bedeutung
CommonName	CN	Name des Objekts
LocalityName	L	Ort
StateOrProvinceName	ST	US-Bundesstaat/Provinz
OrganizationName	O	Organisation
OrganizationalUnitName	OU	Organisatorische Untereinheit
CountryName	C	Staat
StreetAddress	STREET	Straßenname
Email	Email	E-Mail-Adresse

Tabelle 4.4: Komponenten des Distinguished Names (DN)

Der Distinguished Name (DN) besteht aus einer eindeutigen Zusammenstellung der in der Tabelle genannten Einträge.

Mit Hilfe dieser Begriffe kann in einem LDAP-Server der dazugehörige zertifizierte Schlüssel gesucht werden. Auch die Übertragung vorzeitig abgelaufener Zertifikate (Revocation List) geschieht über den Zugriff auf einen DN. Das X.509-Zertifikat aus Abbildung 4.43 enthält zwei DN-Strukturen, je eine für den Herausgeber des Zertifikats (Issuer) und eine für den Subscriber (Subject):

Issuer: CN=GlobalSign Class 2A CA, O=GlobalSign nv-sa, C=BE

Subject: CN=Markus a Campo, C=DE, Email=mac.ac@t-online.de

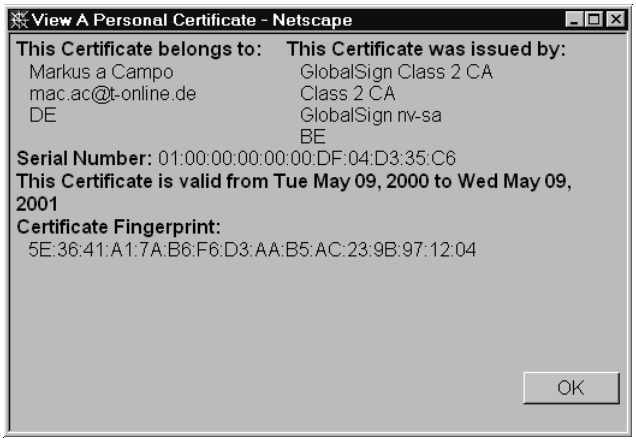

Abb. 4.43: Zertifikat nach X.509