Universität Hamburg

FACHBEREICH WIRTSCHAFTSWISSENSCHAFTEN

Institut für Wirtschaftsinformatik

Diplomarbeit in
Betriebswirtschaftlicher Datenverarbeitung

Prüfer:
Prof. Dr. D. B. Preßmar

Thema:

„Leistungsfähigkeit von kryptografischen Verfahren im Hinblick auf Vertraulichkeit, Verbindlichkeit und Verfügbarkeit"

Abgabetermin: 24. Januar 2001

bearbeitet von:

Björn Buske
Kaiser-Friedrich-Ufer 9
20253 Hamburg

Matr.-Nr.: 4943078
Tel.: 040 - 420 68 53
Fachrichtung: BWL
Fachsemester: 9

Betreuerin:

Miriam O'Shea

Inhalt

Verzeichnis der Tabellen

Verzeichnis der Abbildungen

Verzeichnis der Anmerkungen

Verzeichnis der Abkürzungen

a.a.O.	am angegebenen Ort
Abb.	Abbildung
ACM	Association for Computing Machinery
Anm.	Anmerkung
ANSI	American National Standards Institute
Aufl.	Auflage
bzw.	beziehungsweise
CBC	Cipher Block Chaining
CFB	Cipher Feedback
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
d.h.	das heißt
ECB	Electronic Codebook
ed.	edition (Auflage)
EDI	Electronic Data Interchange
EDIFACT	EDI for Administration, Commerce and Trade
EG	Europäische Union
et al.	et alibi (und anderswo), et alii (und andere)
etc.	et cetera (und so weiter)
f.	folgende (Seite)
ff.	fortfolgende (Seiten)
FIPS	Federal Information Processing Standard
HMD	Handbuch der maschinellen Datenverarbeitung
Hrsg.	Herausgeber

IBM	International Business Machines
IDEA	International Data Encryption Algorithm
IEEE	Institute of Electrical and Electronics Engineers
IPES	Improved Proposed Encryption Standard
ISO	International Organization for Standardization
IT	Informationstechnologie
Jg.	Jahrgang
jr.	junior
KES	Zeitschrift für Kommunikations- und EDV-Sicherheit
MD	Message Digest
MIT	Massachusetts Institute of Technology
NBS	National Bureau of Standards
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OFB	Output Feedback
OSI	Open Systems Interconnection
PES	Proposed Encryption Standard
RACE	Research and Development in Advanced Communication Technologies in Europe
RIPEMD	RACE Integrity Primitives Evaluation Message Digest
RSA	Rivest Shamir Adleman
S.	Seite(n)
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SIAM	Society for Industrial and Applied Mathematics
Tab.	Tabelle
u.	und
vgl.	vergleiche
WISU	Das Wirtschaftsstudium
WWW	World Wide Web
XOR	exclusive or, bitweise exclusiv-oder Verknüpfung
z.B.	zum Beispiel

Verzeichnis der Symbole

A	32-Bit-Variable
$A(C)$	Ersatzalgorithmus zur Entschlüsselung von Nachrichten
a	8-Bit-Teilblock, 32-Bit-Variable
B	Block, 32-Bit-Variable
$B_{i,j}$	i-ter Unterblock nach der j-ten Iteration
B_L	linke Blockhälfte
$B_{L,i}$	linke Blockhälfte nach der i-ten Iteration
B_R	rechte Blockhälfte
$B_{R,i}$	rechte Blockhälfte nach der i-ten Iteration.
b	8-Bit-Teilblock, 32-Bit-Variable
C	Chiffretext, 32-Bit-Variable
C_i	i-ter Chiffretext, i-ter Chiffretextblock
c	8-Bit-Teilblock, 32-Bit-Variable
D	32-Bit-Variable
d	Dechiffrierungsschlüssel des RSA-Algorithmus, 8-Bit-Teilblock,

d'	alternativer RSA-Dechiffrierungsschlüssel
d_i	RSA-Dechiffrierungsschlüssel des i-ten Benutzers
$D(M)$	Dechiffrierungsfunktion
$D_K(M)$	Dechiffrierungsfunktion in Abhängigkeit vom Schlüssel K
E	32-Bit-Variable
e	Chiffrierungsschlüssel des RSA-Algorithmus, 32-Bit-Variable
e_i	RSA-Chiffrierungsschlüssel des i-ten Benutzers
$E(C)$	Chiffrierungsfunktion
$E_K(M)$	Chiffrierungsfunktion in Abhängigkeit vom Schlüssel K
$f(x)$	beliebige mathematische Funktion der Variablen x
g	Primitivwurzel mod p
$H(M)$	Hash-Funktion
h	beliebige ganze Zahl
i	Laufindex
j	Laufindex
K	Schlüssel
K_k	k-ter Schlüssel, k-ter Teilschlüssel
K_t	t-te 32-Bit-Konstante
k	Zufallszahl, Laufindex
L	Länge einer Zahl in Bit
M	Klartext
M'	alternativer Klartext
M_i	i-ter Klartext, i-ter Klartextblock
$M_{i,t}$	t-ter Unterblock des i-ten Klartextblocks
m	natürliche Zahl
n	natürliche Zahl; Modulus des RSA-Algorithmus
n_i	RSA-Modulus des i-ten Benutzers
P_i	i-ter Wert des P-Arrays
p	zufällig gewählte Primzahl zur Berechnung der RSA-Schlüssel
q	zufällig gewählte Primzahl zur Berechnung der RSA-Schlüssel
R	zufällig gewählte Nachricht
r	Teilwert einer DSA- oder ElGamal-Signatur
S	digitale Signatur
$S_{i,j}$	j-ter Eintrag der i-ten S-Box
s	Teilwert einer DSA- oder ElGamal-Signatur
t	Laufindex
u_1	Zwischenwert zur Verifikation von DSA-Signaturen
u_2	Zwischenwert zur Verifikation von DSA-Signaturen
v	Zwischenwert zur Verifikation von DSA-Signaturen
W_t	t-tes 32-Bit-Datenwort
w	Zwischenwert zur Verifikation von DSA-Signaturen
X	beliebige Nachricht
x	beliebige Variable
Y	beliebige Nachricht
y	beliebige Variable
Z	beliebige Nachricht
z	beliebige Variable

1 Einleitung

1.1 Problemstellung

In den letzten Jahren hat der weltweite elektronische Datenaustausch, insbesondere durch den Internet-Boom, in extremer Weise zugenommen. Dies schafft für die Teilnehmer, seien es Betriebe oder Konsumenten, eine Vielzahl neuer Möglichkeiten im Bereich der Kommunikation. Den Teilnehmern erschließen sich Multimedia-Anwendungen, weltweite Recherchemöglichkeiten, aber auch die Möglichkeit der Knüpfung von Geschäftsbeziehungen im elektronischen Medium.

Neben all den vielversprechenden neuen Möglichkeiten stehen aber auch neue Probleme, die damit untrennbar verbunden sind. Wann immer vertrauliche Informationen über unsichere Kanäle versendet werden, oder geschäftliche Transaktionen über unsichere Kanäle abgeschlossen werden, müssen Vorkehrungen getroffen werden, die eine ausreichende Sicherheit gewährleisten können. Moderne kryptografische Verfahren sollen genau das leisten.

Die Wissenschaft hat im Bereich der IT-Sicherheit unter anderem drei wichtige Gesichtspunkte hervorgebracht, die ein sicheres System zu erfüllen hat: Vertraulichkeit, Verbindlichkeit und Verfügbarkeit. In dieser Arbeit sollen kryptografische Verfahren, die der Herstellung von Vertraulichkeit, Verbindlichkeit und Verfügbarkeit dienen, auf ihre Leistungsfähigkeit hin untersucht werden. Dazu wird der aktuelle Bedarf für solche Verfahren in der betrieblichen Umgebung ermittelt und es werden einzelne wichtige Verfahren vorgestellt und bewertet.

1.2 Gang der Untersuchung

Im folgenden Kapitel 2 werden ausgewählte aktuelle Einsatzgebiete für kryptografische Verfahren in der betrieblichen Datenverarbeitung identifiziert und in gebotener Kürze problematisiert.

Anschließend werden im Kapitel 3 die wichtigsten Grundlagen der Kryptografie erläutert, sofern sie für das Verständnis kryptografischer Verfahren von Bedeutung sind. Es werden Anforderungen und Bewertungskriterien für kryptografische Verfahren entwickelt und die zentralen Problembereiche hinsichtlich der Sicherheit und Anwendbarkeit der Verfahren aufgezeigt.

In Kapitel 4 werden die wichtigsten kryptografischen Verfahren zur Gewährleistung der Vertraulichkeit vorgestellt und bezüglich ihrer Sicherheit analysiert.

Kapitel 5 widmet sich den Verfahren, die der Gewährleistung der Verbindlichkeit dienen. Es wird zunächst die rechtliche Problematik erläutert und ein Protokoll für digitale Signaturen vorgestellt, welches den rechtlichen Anforderungen genügt. Daraufhin werden die wichtigsten Verfahren zur Erzeugung digitaler Signaturen erläutert und bezogen auf ihre Sicherheit analysiert.

In Kapitel 6 wird die Problematik der Verfügbarkeit in Verbindung mit kryptografischen Verfahren diskutiert und es werden Verfahren vorgestellt, die zur Gewährleistung der Verfügbarkeit eingesetzt werden können.

In Kapitel 7 werden dann die in den vorhergehenden Kapiteln vorgestellten kryptografischen Verfahren einander zusammenfassend gegenübergestellt und anhand ihrer Leistungsfähigkeit bewertet. Es werden Hinweise zur Auswahl geeigneter Verfahren gegeben und wichtige Gesichtspunkte für die konkrete Implementation aufgezeigt.

Die Arbeit schließt mit einer zusammenfassenden Schlußbetrachtung in Kapitel 8.

2 Anwendungsgebiete kryptografischer Verfahren in der betrieblichen Datenverarbeitung

2.1 Elektronische Märkte

Elektronische Märkte gewinnen immer mehr an Bedeutung. Allgemein umfaßt der Begriff elektronische Märkte das gesamte Spektrum des *E-Commerce*, welches sich in beliebigen Kombinationen zwischen Unternehmen, Konsumenten und Staat abspielen kann. Der weitaus interessanteste Bereich des E-Commerce ist der Business-to-Business-Bereich, nicht nur da er das größte Marktsegment darstellt. Durch elektronische Geschäftsbeziehungen können Unternehmen erhebliche Effizienzsteigerungen erreichen, da durch höhere Übermittlungsgeschwindigkeit und das Wegfallen von Medienbrüchen und doppelten Eingaben die Bearbeitungszeiten und damit die Transaktionskosten stark zurückgehen. [1]

---

[1] Vgl. **Peters, Ralf:** „Technologien des E-Commerce“, in: WISU, 29. Jg. (2000), Heft 7, S. 961 f.

Diese effizienzsteigernde Wirkung zeigt sich beispielsweise auch eindrucksvoll am sogenannten *Electronic Procurement*, der elektronisch unterstützten Beschaffung von Input-Faktoren mit Hilfe des Internet. Im operativen Bereich ergeben sich hier Kostensenkungen, einerseits in den Kosten pro Bestellung, andererseits aber auch in den Prozeßkosten durch eine Beschleunigung des Beschaffungsprozesses. Diese verminderte Reaktionszeit in der Beschaffung verschafft dem Unternehmen über kürzere Auftragsdurchlaufzeiten und mehr Flexibilität eine bessere strategische Wettbewerbsposition. Durch die schlankeren Beschaffungsprozesse werden Freiräume für das Personal im Beschaffungsbereich geschaffen, was sich einerseits in Personalkostensenkungen äußert, andererseits den Verantwortlichen und Mitarbeitern in Materialwirtschaft und Einkauf die Möglichkeit gibt, die Effektivität durch umfassendere Beschaffungsmarktforschung, Entwicklung innovativer Logistiklösungen und genauere Lieferantenbewertungen zu steigern.[2]

Das zentrale Problem für den Erfolg dieser Konzepte ist die Schaffung von Vertrauen. Für die meisten Nutzer sind elektronische Märkte ein neues Medium, mit dem sie nur wenig Erfahrungen gemacht haben, und sie empfinden daher naturgemäß Mißtrauen bezüglich der Sicherheit und Zuverlässigkeit des unbekannten Mediums.[3] Damit ist nicht nur die Sicherheit der übertragenen Informationen vor unbefugtem Zugriff gemeint, sondern auch die Sicherheit vor Betrug und die Sicherheit der Gültigkeit von über das elektronische Medium gemachten Zusagen. Während früher geschäftliche Vereinbarungen meist von Angesicht zu Angesicht ausgehandelt wurden und Zusagen mit Unterschriften auf Vertragsdokumenten bekundet wurden, ist der Kommunikationspartner im elektronischen Markt meist unsichtbar. Die Marktteilnehmer sorgen sich darum, inwiefern man in Anbetracht dieser relativen Anonymität des elektronischen Marktes darauf vertrauen kann, daß der Kommunikationspartner ist, wer er vorgibt zu sein, und inwiefern seine Zusagen bindend sind, so daß er sie nicht später einfach abstreiten kann.[4] Dieses Unbehagen wiegt ebenso schwer, wie die Angst vor unbefugtem Zugriff auf und Mißbrauch von auf öffentlichen Kommu-

[2] Vgl. **Bogaschewsky, Ronald:** „Electronic Procurement – Neue Wege der Beschaffung", in: Elektronischer Einkauf: Erfolgspotentiale, Praxisanwendungen, Sicherheits- und Rechtsfragen, **Hrsg.: Bogaschewsky, Ronald**, Gernsbach 1999, S. 14 ff.

[3] Vgl. **Kuhlen, Rainer:** „Vertrauenssicherung auf elektronischen Märkten", in: WISU, 29. Jg. (2000), Heft 2, S. 221 f.

[4] Vgl. **Bizer, Johann; Miedbrodt, Anja:** „Die digitale Signatur im elektronischen Rechtsverkehr – Deutsches Signaturgesetz und Entwurf der Europäischen Richtlinie", in: Handbuch zum Internetrecht: Electronic Commerce – Informations-, Kommunikations- und Mediendienste, **Hrsg.: Kröger, Detlef; Gimmy, Marc A.**, Berlin et al. 2000, S. 136 ff., sowie **Kuhlen, Rainer**, a.a.O., S. 221 und **Bogaschewsky, Ronald**, a.a.O., S. 36.

nikationswegen versandten sensiblen Daten, wie beispielsweise Zahlungsinformationen oder Vertragsinhalte. Besteht ein signifikantes Maß an Mißtrauen in die Sicherheit und Zuverlässigkeit des Marktes, so führt das zu ausbleibender Akzeptanz der Teilnehmer und der Markt kommt zum Erliegen.[5]

Folglich müssen Unternehmen, um erfolgreich auf elektronischen Märkten agieren zu können, durch entsprechende Sicherheitsmechanismen und eine transparente Sicherheitspolitik Vertrauen untereinander und auch bei den Konsumenten schaffen, denn dieses Vertrauen ist ein kritischer Erfolgsfaktor.[6] Umfragen zufolge ist fehlendes Vertrauen der Konsumenten in die Sicherheit elektronischer Märkte einer der entscheidenden Gründe für die geringeren Wachstumsraten der elektronischen Märkte im Business-to-Consumer-Bereich.[7]

Kryptografische Verfahren können helfen, das notwendige Vertrauen herzustellen, indem einerseits alle Kommunikationen, bei denen vertrauliche Daten übertragen werden mit anerkannt sicheren Verfahren verschlüsselt werden, andererseits verbindliche Aussagen, wie beispielsweise Vertragsabschlüsse, durch sichere digitale Signaturverfahren authentifiziert werden, damit der Urheber zweifelsfrei identifiziert werden kann und die gemachten Aussagen auch bindend auf ihn wirken können.[8]

Dabei bezieht sich dies nicht nur auf das Internet, also hauptsächlich das WWW und E-Mail-Kommunikationen, sondern auch auf Kommunikationen über EDI bzw. EDIFACT oder WebEDI, bei denen die genannten Sicherheitsbedürfnisse ebenfalls bestehen.[9]

2.2 Supply Chain Management

Supply Chain Management ist eine neuere Entwicklung in der Betriebswirtschaftslehre, die in Theorie und Praxis große Aufmerksamkeit erhält, da sie entscheidende Wettbewerbsvorteile verspricht.[10]

---

5 Vgl. **Fuhs, Howard:** „Schutz auf allen Ebenen“, in: NET, 53. Jg. (1999), Heft 11 – 12, S. 48.
6 Vgl. **Kuhlen, Rainer**, a.a.O., S. 221 ff.
7 Vgl. **Kuhlen, Rainer**, a.a.O., S. 221, sowie **Fuhs, Howard**, a.a.O., S. 48.
8 Vgl. **Peters, Ralf**, a.a.O., S. 963 f., sowie **Bogaschewsky, Ronald**, a.a.O., S. 37 ff. und **Fuhs, Howard**, a.a.O., S. 48 f.
9 Vgl. **Peters, Ralf**, a.a.O., S. 962 ff.
10 Vgl. **Werner, Hartmut:** „Supply Chain Management – Partnerschaft zwischen Lieferant und Kunde (I)“, in: WISU, 29. Jg. (2000), Heft 6, S. 813 [2000 a].

Die *Supply Chain* umfaßt alle Aktivitäten, die mit dem Fluß und der Transformation von Gütern vom Rohstoffstadium bis hin zum Endabnehmer verbunden sind, inclusive der zugehörigen Informationsflüsse.[11]

Supply Chain Management ist die Integration dieser Aktivitäten der Unternehmen durch verbesserte Schnittstellen zwischen den Partnern, um gemeinsam einen nachhaltigen Wettbewerbsvorteil zu erreichen.[12]

Dabei werden an den Schnittstellen Verbesserungspotentiale sowohl unternehmensintern, als auch unternehmensextern aufgedeckt und so eine Verbesserung der Güter-, Informations- und Werteflüsse der beteiligten Unternehmen erreicht.[13]

Der angesprochene Informationsfluß entlang der Supply Chain erfordert den Einsatz entsprechender IT-Systeme. Dabei kommen Technologien des E-Commerce, wie beispielsweise das WWW und EDI zum Einsatz.[14] Es werden wichtige Informationen, wie Kundendaten, Verkaufsdaten oder Konkurrenzdaten in einem zentralen Data Warehouse verwaltet und den Partnern in der Supply Chain zweck- und entscheidungsrelevant filtriert zur Verfügung gestellt.[15]

Folglich stellt sich für den Datenaustausch im Supply Chain Management die gleiche Problematik, wie in den oben erläuterten elektronischen Märkten. Es werden vertrauliche Daten über potentiell unsichere Kanäle übertragen und es werden Vertragsabschlüsse und Verhandlungen über das elektronische Medium getätigt. Deshalb ist auch hier der Einsatz von kryptografischen Verfahren zur Verschlüsselung sensibler Daten und zur Authentifizierung von Aussagen durch digitale Signaturen angebracht, um die notwendigen Informationsflüsse abzusichern und damit das Funktionieren des Konzeptes zu ermöglichen.

3 Kryptografische Grundbegriffe

Zweck dieses Kapitels ist, einige grundlegende Definitionen vorzunehmen, sowie die wichtigsten Grundzusammenhänge zu erläutern, die für das Verständnis der späteren

---

[11] Vgl. **Handfield, Robert B.; Nichols, Ernest L., jr.:** Introduction to Supply Chain Management, London et al. 1999, S. 2.
[12] Vgl. ebenda, S. 2.
[13] Vgl. **Werner, Hartmut**, 2000 a, a.a.O., S. 813.
[14] Vgl. **Handfield, Robert B.; Nichols, Ernest L., jr.**, a.a.O., S. 28 ff.
[15] Vgl. **Werner, Hartmut:** „Supply Chain Management – Partnerschaft zwischen Lieferant und Kunde (II)“, in: WISU, 29. Jg. (2000), Heft 7, S. 944 [2000 b].

Untersuchung von kryptografischen Verfahren in Bezug auf die Kriterien Vertraulichkeit, Verbindlichkeit und Verfügbarkeit von Bedeutung sind.

3.1 Grundlegende Definitionen

Kryptografie ist die Wissenschaft von der Sicherung von Nachrichten gegen unbefugte Einflüsse. Dabei sind *Nachrichten* grundsätzlich Inhalte (im Sinne der Datenverarbeitung: Daten) beliebiger Art, die ein beliebiger *Sender* einem beliebigen *Empfänger* mitteilen möchte.[16] *Einflüsse* sind jede Art von Zugriff auf die Nachricht, ob lesend oder ändernd.

Der Kryptografie gegenüber steht die *Kryptoanalyse*. Dies ist die Wissenschaft von der Entzifferung kryptografisch gesicherter Nachrichten ohne Zugriff auf die zur Entschlüsselung notwendigen Informationen. Ihr Ziel ist folglich der *Bruch* kryptografischer Verfahren.[17] Obwohl die Kryptoanalyse auf den ersten Blick der Kryptografie schaden zu wollen scheint, so ist sie doch ein unverzichtbares Hilfsmittel für den Kryptografen bei dem Entwurf kryptografischer Verfahren, da ein solches Verfahren nur als sicher angesehen werden kann, wenn es trotz wissenschaftlich fundierter Bemühungen nicht gebrochen werden konnte. Die Kryptoanalyse stellt also neben dem potentiellen Gegner der Kryptografie auch eine notwendige Kontrollinstanz dar, die es zu schlagen gilt.[18]

Kryptografie und Kryptoanalyse bilden zusammen die *Kryptologie*, diese wiederum ist Teil der Mathematik.[19]

Ein kryptografisches Verfahren schützt generell eine Nachricht durch Verschlüsselung, im folgenden als *Chiffrierung* (englisch: *encryption*) bezeichnet, und darauffolgende Entschlüsselung, im folgenden als *Dechiffrierung* (englisch: *decryption*) bezeichnet. Die unverschlüsselte Nachricht wird *Klartext* (englisch: *plaintext, message*) genannt. Durch Chiffrierung entsteht daraus der *Chiffretext* (englisch: *ciphertext*). Dieser wird durch Dechiffrierung wieder in den Klartext zurückgewandelt.[20] Dieser Zusammenhang wird in Abb. 1 des Anhangs grafisch verdeutlicht. Im folgenden

---

[16] Vgl. **Schneier, Bruce:** Applied Cryptography – Protocols, Algorithms, and Source Code in C, 2nd ed., New York et al. 1996, S. 1.
[17] Vgl. ebenda, S. 1 u. 5.
[18] Vgl. ebenda, S. 7 f.
[19] Vgl. ebenda, S. 1.
[20] Vgl. ebenda, S. 1.

wird in symbolischen Darstellungen entsprechend den englischen Bezeichnungen der Klartext mit *M* (für *message*) und der Chiffretext mit *C* notiert.

Die mathematische Funktion, welche zur Chiffrierung bzw. Dechiffrierung benutzt wird, wird als *kryptografischer Algorithmus* bezeichnet. Oft handelt es sich dabei um zwei getrennte Funktionen, die zueinander in reziproker Beziehung stehen. Im folgenden wird in symbolischen Darstellungen (entsprechend den englischen Bezeichnungen) die Chiffrierungsfunktion als $E(M)$ und die Dechiffrierungsfunktion als $D(C)$ notiert. Dabei gilt $E(M) = C$ und $D(C) = M$ und somit auch $D(E(M)) = M$.[21]

Historisch bestand das Sicherheitsmoment von kryptografischen Algorithmen meist in der Geheimhaltung der Funktionsweise des Algorithmus selber. Diese Art von Algorithmen bezeichnet man als *geheime Algorithmen*. Nach heutigen Maßstäben sind geheime Algorithmen nicht mehr angemessen. Sie können nicht für große oder wechselnde Benutzergruppen verwendet werden, da jedes Mal, wenn ein Benutzer die Gruppe verläßt, ein neuer Algorithmus eingeführt werden muß. Weiterhin muß jede Gruppe ihren eigenen Algorithmus selber entwerfen, da eine Standardisierung konzeptbedingt unmöglich ist. Befindet sich in einer Gruppe kein guter Kryptograf, ist es unmöglich, die Qualität des betreffenden Algorithmus zu überprüfen.[22]

Moderne kryptografische Algorithmen lösen dieses Problem durch die Einführung eines *Schlüssels* (englisch: *key*). Ein Schlüssel ist eine zusätzliche Information, wie z.B. eine Ziffernfolge, die der Algorithmus zur Chiffrierung und Dechiffrierung der Nachricht benötigt. Ohne Kenntnis des Schlüssels ist es trotz Kenntnis des Algorithmus nicht möglich die Nachricht korrekt zu dechiffrieren. Dies wird durch Abb. 2 des Anhangs verdeutlicht.[23] In diesem Fall sollte das gesamte Sicherheitsmoment des Algorithmus in der Geheimhaltung des Schlüssels liegen, der Algorithmus selber kann allgemein bekannt sein. Diese Forderung wurde 1883 von dem niederländischen Philologen AUGUSTE KERCKHOFFS VON NIEUWENHOF postuliert und ist als *Prinzip von KERCKHOFFS* in die Literatur eingegangen.[24] Es ermöglicht die Standardisierung von kryptografischen Algorithmen, z.B. für die Massenproduktion von Softwareprodukten, und die Analyse der Algorithmen hinsichtlich ihrer Sicherheit.[25]

---

21 Vgl. ebenda, S. 2.
22 Vgl. ebenda, S. 3.
23 Vgl. ebenda, S. 3.
24 Vgl. **Oppliger, Rolf:** IT-Sicherheit – Grundlagen und Umsetzung in der Praxis, Braunschweig, Wiesbaden 1997, S. 47.
25 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 3 f.

Im folgenden wird in symbolischen Darstellungen der Schlüssel entsprechend der englischen Bezeichnung mit K notiert. Die Abhängigkeit der Chiffrierungs- und Dechiffrierungsfunktion vom Schlüssel wird durch ein tiefgestelltes K verdeutlicht: $E_K(M)$, bzw. $D_K(C)$. Weitere Ausführungen zur Sicherheit von kryptografischen Algorithmen finden sich in Abschnitt 3.3.

Die Gesamtheit aller zulässigen Werte eines bestimmten Schlüssels wird als *Schlüsselraum* bezeichnet. Ein kryptografischer Algorithmus zusammen mit dem gesamten Schlüsselraum und allen möglichen Klartexten und den zugehörigen Chiffretexten wird als *Kryptosystem* bezeichnet.[26]

Es existiert eine Vielzahl verschiedener kryptografischer Algorithmen. Eine Typisierung der modernen Verfahren wird in Abschnitt 3.5 vorgenommen.

Der Versuch unbefugter Dritter, ein Kryptosystem durch kryptoanalytische Verfahren zu *brechen* – also chiffrierte Nachrichten ohne Kenntnis des Schlüssels zu Dechiffrieren oder den Schlüssel zu ermitteln – wird, ungeachtet der tatsächlichen Intention des Dritten, als *Angriff* bezeichnet, der Dritte selber heißt *Angreifer*.[27] Die möglichen Angriffsstrategien, die einem Kryptoanalytiker zur Verfügung stehen, werden im folgenden Abschnitt 3.2 näher beschrieben.

3.2 Mögliche Angriffe auf Kryptosysteme

3.2.1 Annahmen

Im folgenden werden mögliche Angriffsstrategien für eine Kryptoanalyse erläutert. Dabei wird davon ausgegangen, daß der Angreifer uneingeschränkten Zugriff auf die zwischen Sender und Empfänger verwendeten Kommunikationskanäle hat und daß ihm die Funktionsweise des verwendeten kryptografischen Algorithmus vollständig bekannt ist. Die zweite Annahme geht zurück auf das Prinzip von KERCKHOFFS.[28] Auch wenn in der Realität der Angreifer den verwendeten Algorithmus nicht immer kennt, ist diese Annahme für die theoretische Diskussion und die kritische Überprüfung von Algorithmen auf deren Sicherheit hin dennoch sinnvoll, da ein Algorith-

26 Vgl. ebenda, S. 3 f.
27 Vgl. ebenda, S. 5.
28 Vgl. ebenda, S. 5.

mus, der auch mit detailliertem Wissen über dessen Funktionsweise nicht gebrochen werden kann, ohne dieses Wissen erst recht nicht angreifbar ist.[29]

Grundsätzlich greift ein kryptoanalytischer Angreifer auf mathematische und statistische Werkzeuge, bekannte kryptografische Algorithmen, zuvor abgefangene Chiffretexte desselben Senders und eventuell bereits dechiffrierte Chiffretexte zurück. Von großer Wichtigkeit ist auch ein fundiertes Hintergrundwissen über statistische Eigenschaften und Besonderheiten der in Frage kommenden Sprachen.[30]

Es werden nun zunächst die vier grundlegenden kryptoanalytischen Angriffstypen erläutert. Dann wird auf vier weitere Angriffe eingegangen, die weniger praktikabel, von exotischerer Natur oder nicht mathematisch sind. Die Angriffe werden unterschieden und benannt nach den Informationen, die dem Kryptoanalytiker zur Verfügung stehen.

Anschließend wird auf die Einflußfaktoren für die Komplexität von Angriffen eingegangen und eine Kategorisierung der möglichen Brüche von Kryptosystemen vorgenommen.

3.2.2 Ciphertext-only-Angriff

Dies ist der grundlegendste und offensichtlichste Fall. Der Kryptoanalytiker besitzt die Chiffretexte mehrerer mit dem betreffenden Algorithmus chiffrierter Nachrichten. Seine Aufgabe besteht darin, vorzugsweise den oder die verwendeten Schlüssel, mindestens jedoch die Klartexte von so vielen Nachrichten wie möglich zu ermitteln. Alternativ kann eventuell auch ein Ersatz-Algorithmus $A(C) = M$ entwickelt werden, der mit dem betreffenden Schlüssel chiffrierte Nachrichten dechiffriert, ohne daß der Schlüssel bekannt ist ($A(C)$ ist äquivalent zu $D_K(C)$ für einen bestimmten Schlüssel K). Dieser müßte natürlich für jeden verwendeten Schlüssel getrennt ermittelt werden.[31]

---

[29] Vgl. **Heuser, Ansgar:** „Kryptographie: der Schlüssel zu mehr Datensicherheit in der Informationstechnik“, in: HMD – Theorie und Praxis der Wirtschaftsinformatik, 33. Jg. (1996), Heft 190, S. 9 f.
[30] Vgl. **Oppliger, Rolf**, a.a.O., S. 49 f.
[31] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 5 f.

3.2.3 Known-plaintext-Angriff

In diesem Fall sind dem Kryptoanalytiker neben den Chiffretexten mehrerer Nachrichten auch die dazugehörigen Klartexte bekannt. Seine Aufgabe besteht darin, den oder die verwendeten Schlüssel, oder mindestens entsprechende Ersatzalgorithmen *A(C)* zur Entschlüsselung weiterer mit denselben Schlüsseln chiffrierter Nachrichten abzuleiten.[32]

3.2.4 Chosen-plaintext-Angriff

Dem Kryptoanalytiker war es möglich, selbst erstellte Klartexte in das gegnerische Kommunikationssystem einzuschleusen und chiffrieren zu lassen und die entsprechenden Chiffretexte abzufangen. Die Aufgabe des Kryptoanalytikers ist dieselbe, wie beim Known-plaintext-Angriff.[33]

Diese Angriffsvariante ist mächtiger als die vorherige, da der Kryptoanalytiker Klartexte entwerfen kann, die ihm helfen können, mehr Informationen über den Schlüssel abzuleiten, indem sie z.B. im Chiffretext Strukturen offenlegen.[34]

3.2.5 Adaptive-chosen-plaintext-Angriff

Dies ist eine erweiterte Form des Chosen-plaintext-Angriffs. Während im obigen Fall der Kryptoanalytiker eventuell nur einmal die Chance hat, einen Text zur Chiffrierung einzuschleusen, steht ihm hier diese Möglichkeit wiederholt zur Verfügung. Somit kann er weitere, auf Basis der Ergebnisse der Analyse der vorherigen Klartext-Chiffretext-Paare gezielt abgeänderte Klartexte einschleusen und damit seine Analysemöglichkeiten beträchtlich erweitern.[35]

Diese Angriffsform ist die mächtigste der vier klassischen kryptoanalytischen Angriffe.

3.2.6 Chosen-ciphertext-Angriff

Ein seltenerer Fall ist der, daß der Kryptoanalytiker einen Text erstellt, diesen als zu entschlüsselnden Chiffretext in das System einschleust und Zugriff auf das Ergebnis der Dechiffrierung hat. Eine solche Situation kann z.B. auftreten, wenn der Kryptoa-

32 Vgl. ebenda, S. 6.
33 Vgl. ebenda, S. 6.
34 Vgl. ebenda, S. 6.
35 Vgl. ebenda, S. 6.

nalytiker im Besitz einer nicht manipulierbaren Dechiffriermaschine ist und in diese Daten einspeist. Seine Aufgabe ist hier die Ermittlung des Schlüssels.[36]

3.2.7 Chosen-key-Angriff

In dieser Variante hat der Kryptoanalytiker Informationen über die Beziehungen oder Abhängigkeiten zwischen verschiedenen Schlüsseln und ist dadurch eventuell in der Lage, einen Schlüssel abzuleiten. Dieser Angriffstyp ist obskur, wenig praktikabel und dürfte nur in seltenen Fällen von Bedeutung sein.[37]

3.2.8 Brute-force-Angriff

Dies ist eine spezielle Form des Ciphertext-only- oder des Known-plaintext-Angriffs. Der Kryptoanalytiker probiert einfach sämtliche Schlüssel des Schlüsselraums daraufhin aus, ob sie bei der Dechiffrierung einen sinnvollen bzw. den bekannten Klartext ergeben. Während dieser Angriffstyp bei kleinem Schlüsselraum durchaus sinnvoll sein kann, wird er um so unpraktikabler, je größer der Schlüsselraum, und damit die Anzahl möglicher Schlüssel ist.[38]

3.2.9 Nicht-kryptoanalytische Angriffe

Oftmals die besten Möglichkeiten, ein Kryptosystem zu brechen sind die Angriffe, bei denen dem Angreifer der Schlüssel durch Methoden wie Erpressung, Folter, Bestechung oder Diebstahl in die Hände fällt. Diese Angriffe sind in der Regel sehr mächtig und ihnen kann auf kryptografischem Wege nicht vorgebeugt werden.[39]

3.2.10 Komplexität von Angriffen auf Kryptosysteme

Die Komplexität eines Angriffes auf ein Kryptosystem wird von drei Einflußfaktoren bestimmt und kann folglich anhand dieser Faktoren gemessen werden. Diese sind im einzelnen:[40]

1. **Datenmäßige Komplexität:** Ein Angriff ist um so komplexer, je mehr Eingabedaten für eine erfolgreiche Durchführung benötigt werden.

---

36 Vgl. ebenda, S. 6 f.
37 Vgl. ebenda, S. 7.
38 Vgl. ebenda, S. 8 u. 152 f.
39 Vgl. ebenda, S. 7.
40 Vgl. ebenda, S. 8 f.

2. **Rechnerische Komplexität:** Ein Angriff ist um so komplexer, je mehr Rechenzeit für eine erfolgreiche Durchführung benötigt wird.
3. **Speicherbedarf:** Ein Angriff ist um so komplexer, je mehr Speicherkapazität für eine erfolgreiche Durchführung benötigt wird.

Es kann zwischen den drei Faktoren zu Wechselwirkungen kommen, z.B. kann es möglich sein, die rechnerische Komplexität eines Angriffes auf Kosten eines erhöhten Speicherbedarfs zu reduzieren.[41]

Die Größenordnung der Komplexität wird in Potenzen angegeben. Eine rechnerische Komplexität von 2^{128} beispielsweise bedeutet, daß zur Durchführung des betreffenden Angriffes 2^{128} Rechenoperationen erforderlich sind.[42]

3.2.11 Kategorien des Bruches von Kryptosystemen

Ergebnis eines erfolgreichen Angriffes ist der *Bruch* eines Kryptosystems. Nach LARS KNUDSEN existieren, mit abnehmender Schwere des Bruches, folgende vier Kategorien:[43]

1. **Totaler Bruch:** Dem Kryptoanalytiker gelingt es, den verwendeten Schlüssel K zu ermitteln, so daß er mit Hilfe der Dechiffrierungsfunktion $D_K(C)$ jede mit diesem Schlüssel chiffrierte Nachricht dechiffrieren kann.
2. **Globale Deduktion:** Der Kryptoanalytiker findet, ohne den Schlüssel K zu kennen, einen Ersatzalgorithmus $A(C)$, welcher äquivalent zu $D_K(C)$ ist, und ist somit in der Lage, alle mit K verschlüsselten Nachrichten zu dechiffrieren.
3. **Lokale Deduktion:** Der Kryptoanalytiker schafft es, den Klartext zu einem abgefangenen Chiffretext zu ermitteln.
4. **Informations-Deduktion:** Der Kryptoanalytiker gewinnt bestimmte Informationen, beispielsweise über den Schlüssel oder über den Klartext eines abgefangenen Chiffretexts.

---

41 Vgl. ebenda, S. 9.
42 Vgl. ebenda, S. 9.
43 Vgl. ebenda, S. 8.

3.3 Sicherheit von Kryptosystemen

Unter dem Begriff *Sicherheit* von Kryptosystemen versteht man die Sicherheit des verwendeten Algorithmus vor kryptoanalytischen Angriffen. Generell ist ein Kryptosystem um so sicherer, je schwerer es zu brechen ist.[44]

Ein Kryptosystem wird als *bedingungslos sicher* bezeichnet, wenn es auch mit unendlich großen rechnerischen Ressourcen unmöglich zu brechen ist, ganz egal wie viele Chiffretexte der Kryptoanalytiker auch besitzen mag. [45] In diesem Fall läßt der Chiffretext keinerlei verwertbare Informationen über den Klartext durchblicken. Das Kryptosystem bietet *perfekte Geheimhaltung*.[46] Nach SHANNON ist dies nur möglich, wenn die Anzahl möglicher Schlüssel mindestens so groß ist, wie die Anzahl möglicher Klartexte. Das wiederum bedeutet, daß der Schlüssel mindestens so lang sein muß, wie der Klartext und kein Schlüssel mehr als einmal benutzt werden darf. Es existiert kein realisierbares Kryptosystem (außer einem One-time Pad, das jedoch unpraktikabel ist, vgl. dazu Anm. 1 des Anhangs), das diese Bedingung erfüllt. Alle bekannten algorithmenbasierten Systeme können prinzipiell durch einen Brute-force-Angriff gebrochen werden.[47]

In der Praxis werden somit andere Sicherheitsmaßstäbe notwendig. Daher ist das Ziel der Kryptografie die Entwicklung von Algorithmen, deren Bruch *rechnerisch undurchführbar* ist. Das bedeutet, daß auch der beste bekannte Angriff auf das Kryptosystem so komplex sein muß, daß er sowohl mit heutigen, als auch zukünftig verfügbaren Rechnerkapazitäten nicht durchführbar ist. Ein Algorithmus mit dieser Eigenschaft wird als *rechnerisch sicher* bezeichnet.[48]

Ein Kryptosystem gilt als *praktisch sicher*, wenn der beste bekannte Angriff nicht einfacher ist, als ein Brute-force-Angriff. Ein Brute-force-Angriff wird rechnerisch undurchführbar, wenn der Schlüsselraum hinreichend groß ist und die verwendeten Schlüssel zufällig aus dem Schlüsselraum ausgewählt werden.[49] Ein großer Schlüsselraum ist somit eine notwendige, aber keine hinreichende Bedingung für die Sicherheit eines Kryptosystems.[50]

---

[44] Vgl. ebenda, S. 8.
[45] Vgl. ebenda, S. 8.
[46] Vgl. **Oppliger, Rolf**, a.a.O., S. 52 f.
[47] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 8 u. 235.
[48] Vgl. ebenda, S. 8.
[49] Vgl. **Oppliger, Rolf**, a.a.O., S. 53.
[50] Vgl. ebenda, S. 49.

3.4 Anforderungen an kryptografische Verfahren

3.4.1 Die Leistungsfähigkeit kryptografischer Verfahren

Die Leistungsfähigkeit eines kryptografischen Verfahrens ist für den praktischen Einsatz von entscheidender Bedeutung. Allgemein formuliert ist ein Verfahren erwünscht, welches bei akzeptablem Aufwand für Implementation und Nutzung hinreichende Sicherheit vor kryptoanalytischen Angriffen bietet.[51]

In der Literatur zur Sicherheit von IT-Systemen (Sicherheit im weiteren Sinne) wurden zur Beurteilung der Leistungsfähigkeit von Sicherheitsmaßnahmen unter anderem die drei Kriterien *Vertraulichkeit*, *Verbindlichkeit* und *Verfügbarkeit* entwickelt. In dieser Arbeit sollen kryptografische Verfahren auf ihre Leistungsfähigkeit in bezug auf diese Kriterien hin untersucht werden. Dazu werden diese Kriterien im folgenden definiert. Anschließend werden die gängigen kryptografischen Anforderungen an kryptografische Algorithmen erläutert.

3.4.2 Vertraulichkeit

Ein sicheres IT-System gewährleistet die Vertraulichkeit von Informationen. Das bedeutet, daß die in dem System gespeicherten, verarbeiteten oder übertragenen Informationen nur berechtigten Personen zugänglich und somit vor jeglichem unberechtigten Zugriff geschützt sind.[52]

Dieses Kriterium betrifft die Chiffrierverfahren. Es lassen sich hier sämtliche Anforderungen subsumieren, welche sich mit der Sicherheit von Kryptosystemen (Sicherheit im engeren Sinne) beschäftigen.

3.4.3 Verbindlichkeit

Werden über ein IT-System vertragliche Transaktionen oder ähnliches abgewickelt, so muß das System die Verbindlichkeit garantieren. Das bedeutet, daß das System mit entsprechenden Verfahren sicherzustellen hat, daß die übertragenen Daten korrekt und unverändert ankommen, daß sich der Absender einer Nachricht eindeutig, fälschungssicher und nachweislich identifizieren kann und daß der Empfang einer

---

[51] Vgl. ebenda, S. 47.
[52] Vgl. ebenda, S. 11.

Nachricht nachweisbar ist.[53] In der Terminologie der ISO heißt dieses Kriterium *nonrepudiation*, was sich darauf bezieht, daß Autor und Empfänger einer Nachricht die Autorenschaft bzw. den Empfang nicht glaubhaft abzustreiten in der Lage sein dürfen.[54]

Im kryptografischen Sinn betrifft dieses Kriterium die Verfahren digitaler Signaturen und der Empfangsbestätigung.

3.4.4 Verfügbarkeit

Ein IT-System muß die Verfügbarkeit von Informationen gewährleisten. Das bedeutet, daß bei einem Zugriff die geforderte Information innerhalb einer angemessenen Zeitspanne in geforderter Form und Qualität zur Verfügung stehen muß.[55]

Im Zusammenhang mit kryptografischen Verfahren entspricht dies der Forderung nach Verfahren, die einen möglichst geringen Zeitaufwand für Chiffrierung und Dechiffrierung erfordern und die sicherstellen, daß keine Veränderung des Klartextes stattfindet. Die Verfahren müssen also gewährleisten, daß Veränderungen durch Übertragungsfehler oder Manipulationen Dritter vom Empfänger erkannt werden können.

3.4.5 Kryptografische Anforderungen an Kryptosysteme

Die Kryptografie hat selber eine Reihe von Anforderungen an kryptografische Algorithmen hervorgebracht. Dies sind neben dem grundlegenden Prinzip von KERCKHOFFS, das in Abschnitt 3.1 erläutert wurde, vor allem die aus dem Werk des Informationstheoretikers CLAUDE E. SHANNON hervorgegangenen Kriterien.[56]

SHANNON hat zwei Konzepte für den Entwurf von Algorithmen vorgeschlagen, die sich vorwiegend an die Sicherheit vor kryptoanalytischen Angriffen richten:[57]

1. **Konfusion:** Der funktionale Zusammenhang zwischen Klartext, Chiffretext und dem Schlüssel sollte so komplex wie möglich sein.
2. **Diffusion:** Jedes Zeichen des Chiffretextes sollte von dem gesamten Schlüssel und darüber hinaus von so vielen Zeichen des Klartextes wie möglich ab-

---

[53] Vgl. **Kersten, Heinrich:** Einführung in die Computersicherheit, München, Wien 1991, S. 52.
[54] Vgl. ebenda, S. 84 f.
[55] Vgl. **Oppliger, Rolf**, a.a.O., S. 11.
[56] Vgl. ebenda, S. 47.
[57] Vgl. ebenda, S. 48.

hängig sein, damit die statistischen Eigenschaften des Klartextes (z.B. die Häufigkeiten bestimmter Worte in einer Sprache, Redundanz einer Sprache) möglichst gut maskiert werden.

Hinreichende Konfusion und Diffusion tragen folglich zur rechnerischen Sicherheit eines Kryptosystems bei.

Darüber hinaus postulierte SHANNON fünf Kriterien zur Beurteilung von Kryptosystemen:[58]

1. **Höhe der Geheimhaltung:** Dieses Kriterium meint allgemein die Sicherheit eines Kryptosystems, wie unter Abschnitt 3.3 diskutiert.
2. **Schlüssellänge:** Schlüssel müssen in der Regel gespeichert und unter Umständen übertragen werden, so daß sie nicht beliebig lang sein dürfen.
3. **Komplexität der Chiffrierungs- und Dechiffrierungsoperationen:** Die bei der Chiffrierung und Dechiffrierung auszuführenden Rechenoperationen dürfen nicht zu komplex werden, damit sie mit der zur Verfügung stehenden Rechenleistung durchführbar sind. Dieses Kriterium tritt mit wachsenden Rechenleistungen mehr und mehr in den Hintergrund.
4. **Fehlerausbreitung:** Übertragungsfehler sollten sich nur lokal auf den Klartext auswirken, er sollte nicht durch ein falsches Bit an einer bestimmten Stelle vollständig unlesbar werden.
5. **Nachrichtenexpansion:** In der Regel ist es unerwünscht, wenn sich Nachrichten bei der Chiffrierung übermäßig aufblähen.

Vor dem Hintergrund des Abschnitts 3.3 ist bei den Kriterien 2. und 3. zu beachten, daß in Anbetracht der Sicherheit – und damit der Erfüllung des Kriteriums 1. – eine gewisse Mindestlänge der Schlüssel und eine Mindestkomplexität der Chiffrierung und Dechiffrierung nicht unterschritten werden kann.

[58] Vgl. ebenda, S. 48 f.

3.5 Typisierung kryptografischer Algorithmen

3.5.1 Symmetrische Algorithmen

3.5.1.1 Grundlagen

Es gibt zwei Grundtypen schlüsselbasierter Algorithmen: *symmetrische* und *asymmetrische Algorithmen*.[59] Bei *symmetrischen Algorithmen* sind die Schlüssel zur Chiffrierung und Dechiffrierung identisch oder können zumindest leicht voneinander abgeleitet werden. Dies setzt voraus, daß Sender und Empfänger sich zunächst über einen anderweitig gesicherten Kanal auf einen Schlüssel einigen müssen, bevor sie sicher kommunizieren können.[60] Die Sicherheit des Kryptosystems liegt im Schlüssel, dieser ist unbedingt geheimzuhalten.[61]

Symmetrische Algorithmen sind vergleichsweise leicht zu implementieren, sowohl in Hardware-, als auch in Softwarelösungen. Sie erlauben es, Daten in sehr hoher Geschwindigkeit zu chiffrieren und dechiffrieren. Der Durchsatz liegt maximal bei mehreren Hundert Megabit pro Sekunde. [62]

Generell sind mit symmetrischen Kryptosystemen drei Probleme verbunden:[63]

1. Die Schlüssel müssen auf absolut sicherem Wege ausgetauscht werden, da sie ebenso wertvoll sind, wie die zu übertragenden Informationen, denn sie ermöglichen den direkten Zugang zu den Informationen. Bei Kommunikationen über große Distanzen ist dies eine äußerst komplexe Aufgabe. Oftmals werden Schlüssel per Kurier von Hand verbracht.

2. Wenn ein Schlüssel Dritten in die Hände fällt, können diese nicht nur jeglichen damit chiffrierten Nachrichtenverkehr dechiffrieren, sondern sich auch als Sender ausgeben und gefälschte Nachrichten verschicken.

3. Optimalerweise sollte zwischen jedem möglichen Paar von Kommunikationsteilnehmern in einem Netzwerk ein eigener Schlüssel vereinbart sein. Bei einem Netzwerk mit n Benutzern führt das zu $n * (n - 1) / 2$ Schlüsseln. Bei $n = 10$ ergäbe dies 45, bei $n = 100$ schon 4950 Schlüssel. Folglich wird bei

---

59 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 4.
60 Vgl. **Buchmann, Johannes:** Einführung in die Kryptographie, Berlin et al. 1999, S. 56 f.
61 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 4.
62 Vgl. **Heuser, Ansgar**, a.a.O., S. 10.
63 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 29.

Netzwerken mit vielen Benutzern die Schlüsselverwaltung extrem kompliziert für alle Beteiligten.

Die symmetrischen Algorithmen werden weiter unterteilt in zwei Untergruppen, die sogenannten *Stromchiffren* und die *Blockchiffren*.[64] Diese werden im folgenden erläutert

3.5.1.2 Stromchiffren

Stromchiffren verarbeiten den Klartext Bit- oder Byteweise (Zeichenweise).[65] Dazu wird auf dem Schlüssel basierend ein *Schlüsselstrom* generiert, für jedes zu chiffrierende Bit oder Zeichen des Klartextes je ein Bit oder Zeichen des Schlüsselstroms. Mit diesem Schlüsselstrom wird der Klartext nun Bit für Bit bzw. Zeichen für Zeichen verknüpft, woraus sich der Chiffretext ergibt. Die Generatorfunktion für den Schlüsselstrom ist so gestaltet, daß sie einen Strom erzeugt, der zufällig aussieht, jedoch jederzeit exakt reproduziert werden kann, vorausgesetzt der korrekte Schlüssel wurde eingegeben. Zur Dechiffrierung wird mit Hilfe des Schlüssels der korrekte Schlüsselstrom reproduziert. Nun kann durch simples Umkehren der Verschlüsselungsoperationen der Klartext aus dem Chiffretext erzeugt werden.[66] Folglich funktionieren Stromchiffren in dieser Form ausschließlich auf Basis von SHANNON's Konzept der Konfusion. Einige Rückkopplungsmethoden können jedoch auch Diffusion hinzufügen.[67]

3.5.1.3 Blockchiffren

Blockchiffren verarbeiten den Klartext im Gegensatz zu Stromchiffren nicht Bitweise, sondern Blockweise.[68] Dabei bilden sie Klartextblöcke fester Länge n (in Bit) auf Chiffretextblöcke derselben Länge ab.[69] Falls die Länge des Klartextes nicht glatt durch n teilbar ist, wäre der letzte Block zu kurz. In diesem Falle wird er auf die Länge n aufgefüllt, z.B. mit Nullbits, einem bestimmten Bitmuster oder zufälligen Bits. Diese Füllbits werden bei der Dechiffrierung wieder entfernt.[70] Eine typische Blockgröße ist z.B. 64 Bit. In der einfachsten Implementation bearbeitet eine Block-

[64] Vgl. ebenda, S. 4.
[65] Vgl. ebenda, S. 4.
[66] Vgl. ebenda, S. 197 f.
[67] Vgl. ebenda, S. 237.
[68] Vgl. ebenda, S. 4.
[69] Vgl. **Buchmann, Johannes**, a.a.O., S. 62.
[70] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 190 f.

chiffre die Klartextblöcke voneinander unabhängig, indem sie jeden Block nach demselben Muster mit dem Schlüssel verknüpft. In diesem Fall ergeben identische Klartextblöcke jedoch immer identische Chiffretextblöcke. Aus diesem Grund wird diese Ausprägung der Blockchiffren als *Electronic-Codebook-Modus*, oder kurz *ECB-Modus* bezeichnet.[71] Aufgrund dieser Eigenschaft werden jedoch bestimmte statistische Informationen über Regelmäßigkeiten im Klartext nicht maskiert, was einem Kryptoanalytiker eine Angriffsfläche bietet. Außerdem ist es möglich, unbemerkt die Reihenfolge von Chiffretextblöcken zu verändern oder Blöcke, die mit demselben Schlüssel chiffriert wurden, einzufügen oder zu wiederholen.[72]

Um dieses Problem zu beheben, existieren drei weitere Modi für Blockchiffren, bei denen die einzelnen Blöcke durch Rückkopplungen voneinander abhängig gemacht werden.[73]

Die nächstkomplexere Form einer Blockchiffre nach dem ECB-Modus ist der *Cipher-Block-Chaining-* bzw. *CBC-Modus*. Hier wird eine Rückkopplung zwischen den Blöcken erreicht, indem ein Klartextblock vor der Chiffrierung mit dem Chiffretext des vorhergehenden Blocks per XOR verknüpft wird. Dadurch wird der Chiffretext eines Blocks von allen vorherigen Blöcken abhängig,[74] es werden also Regelmäßigkeiten im Klartext verdeckt.[75] Außerdem wird so das Einfügen, Wiederholen oder Vertauschen von Blöcken vereitelt, da in diesem Falle die Nachricht unlesbar wird.[76] Problematisch ist, daß für den ersten zu chiffrierenden Block das Ergebnis hier das gleiche ist, wie im ECB-Modus. Außerdem werden zwei Nachrichten mit identischem Anfang auch bis zum ersten verschiedenen Block den identischen Chiffretext ergeben. Dieses Problem wird im Allgemeinen dadurch behoben, daß vor dem ersten Klartextblock eine Art Dummyblock, der sogenannte *Initialisationsvektor*, eingefügt wird. Er muß bei jeder zu chiffrierenden Nachricht unterschiedlich sein und kann damit aus Zufallsdaten oder einem Timestamp oder ähnlichen Daten bestehen. Bei der Dechiffrierung braucht der Initialisationsvektor nicht mit entschlüsselt zu wer-

---

71 Vgl. ebenda, S. 189 f.
72 Vgl. **Buchmann, Johannes**, a.a.O., S. 65.
73 Vgl. ebenda, S. 65 ff.
74 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 193.
75 Vgl. ebenda, S. 196.
76 Vgl. **Buchmann, Johannes**, a.a.O., S. 65 f. u. 68.

den, nur der entsprechende Chiffretextblock ist zur Dechiffrierung des nächsten Blocks notwendig.[77]

Eine interessante Variante des CBC-Modus besteht darin, ein laufendes XOR aller Klartextblöcke zu bilden und das Ergebnis mit dem letzten Klartextblock vor der Chiffrierung per XOR zu verknüpfen. Dadurch wirkt sich eine Änderung des Chiffretextes an irgendeiner Stelle auf das Dechiffrierungsergebnis des letzten Blocks aus. Ist dieser letzte Block eine Prüfsumme irgendeiner Art oder auch einfach nur ein konstanter Wert, so kann hierdurch sehr einfach eine Integritätsprüfung der Nachricht vorgenommen werden. Diese Variante des CBC-Modus wird *Cipher Block Chaining with Checksum* bzw. *CBCC-Modus* genannt.[78]

Dem CBC-Modus sind zwei Probleme inhärent, die in manchen Situationen von Bedeutung sein können. Zunächst ist dies die Fehlerausbreitung. Ein falsches Bit in einem Block des Chiffretexts führt dazu, daß der betreffende Block des dechiffrierten Klartextes unlesbar wird und der nächste Block ein falsches Bit an der entsprechenden Stelle aufweist. Alle weiteren Blöcke werden fehlerfrei entschlüsselt. Bei *Synchronisationsfehlern*, das sind Situationen, in denen während der Übertragung Bits verlorengegangen oder hinzugekommen sind, ist das Ergebnis noch schlimmer. In diesem Fall wird die Nachricht ab der Stelle des Fehlers nicht mehr korrekt dechiffriert, da jeder Block um die entsprechende Anzahl Bits verschoben ist. Das kann durch Einrahmung der einzelnen Blöcke in Blockanfangs- und Blockendesequenzen vermieden werden.[79] Der zweite Problembereich erschließt sich aus dem blockorientierten Ansatz, der voraussetzt, daß immer vollständige Blöcke verarbeitet werden können. Das bedeutet bei der Chiffrierung von laufenden Datenübertragungen, daß die Daten gepuffert werden müssen, bis ein Block voll ist. Bei Übertragungen mit geringem Durchsatz, wie z.B. der Eingabe von Benutzernamen und Passwort an einem Terminal, kann das fatal sein, da die Eingaben langsam erfolgen und eventuell zu kurz sind. Hier ist es wichtig, daß jedes Zeichen einzeln verschlüsselt übertragen werden kann. Für solche Zwecke ist der CBC-Modus nicht geeignet.[80] Die beiden folgenden Modi versuchen, diese Probleme zu beseitigen.

---

77 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 194.
78 Vgl. ebenda, S. 207 f.
79 Vgl. ebenda, S. 195 f.
80 Vgl. ebenda, S. 200.

Der *Cipher-Feedback-* bzw. *CFB-Modus* erlaubt die Chiffrierung von Dateneinheiten, die kleiner sind, als die Blockgröße, indem die Blöcke jeweils in Unterblöcke unterteilt werden. Diese können dann einzeln verschlüsselt werden. Man spricht dabei von einem *n*-Bit-CFB-Modus, wobei *n* die *Feedbacklänge*, das ist die Länge der Unterblöcke, darstellt. Ein 8-Bit-CFB-Mode mit einer Blockgröße von 64 Bit kann somit trotz der Blockgröße von 8 Byte je ein Byte (z.B. ein ASCII-Zeichen) einzeln verschlüsseln. An diesem Zahlenbeispiel läßt sich die Funktionsweise einfach erklären. Anstatt immer ausschließlich den Chiffretext des letzten Blockes vollständig zu puffern und mit dem nächsten vollständigen Klartextblock zu verknüpfen, werden die Chiffretextdaten abschnittweise in ein Schieberegister von der Länge der Blockgröße geladen. Im Beispiel ist das also ein 8-Byte-Schieberegister. Zunächst wird das Schieberegister mit dem Chiffretext des Initialisationsvektors gefüllt. Das erste Klartextbyte wird dann mit dem ältesten Byte im Schieberegister verknüpft und das Ergebnis verschlüsselt. Das neue Chiffretext-Byte wird in das Schieberegister geladen, wodurch das älteste Chiffretext-Byte aus dem Register gelöscht wird. Nach diesem Schema wird dann byteweise weiter fortgefahren. Damit stellt sich das Problem der blockweisen Verschlüsselung nicht mehr, eine Blockchiffre im CFB-Modus kann auch Login-Informationen oder andere Tastatureingaben, sowie langsame Übertragungskanäle verschlüsseln.[81]

Die unterschiedliche Implementation hat jedoch auch Auswirkungen auf die Fehlerausbreitung. Im CFB-Modus beeinflußt ein falsches Bit im Chiffretext die Dechiffrierung so lange, bis es das Schieberegister einmal durchlaufen hat. Das bedeutet, daß bei einem *n*-Bit CFB-Modus mit einer Blockgröße von *m* der fehlerhafte Unterblock selber, sowie maximal die darauffolgenden m/n Unterblöcke (insgesamt also maximal $m/n + 1$ Unterblöcke), fehlerhaft dechiffriert werden und folglich unlesbar sind.[82] Der CFB-Modus ist selbstsynchronisierend, das heißt, daß Synchronisationsfehler die Dechiffrierung nur so lange beeinflussen, wie sie sich im Schieberegister befinden. Ein Synchronisationsfehler führt im Beispiel also zu acht falsch dechiffrierten Bytes.[83] Das gilt freilich nur, wenn ganze Unterblöcke fehlen oder hinzuge-

---

81 Vgl. ebenda, S. 200 f.
82 Vgl. **Knudsen, Lars R.:** „Contemporary Block Ciphers“, in: Lectures on Data Security – Modern Cryptology in Theory and Practice, **Hrsg.: Damgård, Ivan B.**, Berlin et al. 1999, S. 107.
83 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 201 f.

kommen sind. Um 1-Bit-Synchronisationsfehler behebbar zu machen, müßte man einen 1-Bit-CFB-Modus benutzen.[84]

Beim *Output-Feedback-* bzw. *OFB-Modus* wird ähnlich vorgegangen, wie beim CFB-Modus. Der Unterschied besteht darin, daß das Schieberegister nicht mit den letzten erzeugten Chiffretextabschnitten gefüllt wird, sondern mit den letzen Schlüsselsequenzen. Es wird also das Schieberegister mit dem Initialisationsvektor gefüllt und der Inhalt mit dem Schlüssel verknüpft. Das neue Ergebnis wird in das Schieberegister geladen, und so weiter. So erzeugt der Algorithmus mit Hilfe des Schieberegisters praktisch einen Schlüsselstrom, der mit sich selber immer wieder verknüpft wird und von Klartext und Chiffretext unabhängig ist. Dieser Schlüsselstrom ist folglich eine Funktion des Schlüssels und des Initialisationsvektors. Er kann dann mit dem Klartext per XOR verknüpft werden und ergibt so den Chiffretext. Man kann den OFB-Modus also als eine Implementation einer Blockchiffre als Stromchiffre betrachten.[85]

Ein Vorteil dieser Vorgehensweise ist, daß der Schlüsselstrom bereits vorweg generiert und abgespeichert werden kann, um einen eventuell später hinzukommenden Klartext zu chiffrieren. Eine Fehlerausbreitung existiert nicht, ein falsches Bit im Chiffretext führt zu einem korrespondierenden falschen Bit im dechiffrierten Klartext. Das macht den OFB-Modus besonders interessant für Anwendungen, bei denen einzelne Bitfehler toleriert werden können, Blockfehler jedoch nicht, wie beispielsweise Audio- oder Videoströme. Synchronisationsfehler hingegen sind fatal. Ein Audio- oder Video-Übertragungssystem beispielsweise, welches einen Algorithmus im OFB-Modus verwendet, muß eine Möglichkeit vorsehen, einen Synchronisationsverlust zu erkennen und beide Enden der Kommunikation mit einem identischen Initialisationsvektor neu zu initialisieren. Die Unabhängigkeit des Schlüsselstroms von Klartext und Chiffretext bringt jedoch ein Sicherheitsproblem mit sich. Der Schlüsselstrom stellt nämlich eine zyklische Permutation des Initialisationsvektors dar. Diese hat bei einem n-Bit-OFB-Modus mit der Blockgröße m eine mittlere Periode von $2^m - 1$ für den Fall, daß n gleich m ist. Nach dieser Anzahl von Blöcken wiederholt sich die Folge. Dies ist bei einer Blockgröße von 64 Bit ein ausreichend großer Wert. Im Fall $n < m$ jedoch fällt die mittlere Periode der Permutation auf ca. $2^{m/2}$, was bei einer Blocklänge von 64 Bit nicht lang genug ist. Folglich kann der

[84] Vgl. ebenda, S. 209.
[85] Vgl. ebenda, S. 203 f.

OFB-Modus nur empfohlen werden, wenn die Feedbacklänge gleich der Blockgröße ist.[86]

Allen Modi von Blockchiffren ist gemeinsam, daß sie sich zur Chiffrierung beider Konzepte von Shannon – der Konfusion, sowie der Diffusion – bedienen.[87]

Die Auswahl eines konkreten Modus ist abhängig von der zu bewältigenden Aufgabe. Generell läßt sich feststellen, daß der CBC-Modus zur Verschlüsselung von gespeicherten Daten oder am Stück verschickten Nachrichten, wo Synchronisationsfehler kaum auftreten, am besten geeignet ist.[88] Für die Verschlüsselung von Datenströmen sind CFB und OFB besser geeignet, da sie nicht auf das Vorhandensein eines vollständigen Blocks angewiesen sind, um zu funktionieren und so z.B. zeichenweise Verschlüsselung ermöglichen.

CFB ist dabei für Anwendungen gut geeignet, die eine gewisse Unempfindlichkeit gegenüber Synchronisationsfehlern erfordern, bei denen aber die Fehlerausbreitung bei falsch übertragenen Bits nicht von großer Bedeutung ist. Der OFB-Modus ist bei Datenströmen zu empfehlen, die sehr empfindlich auf eine hohe Fehlerausbreitung von falschen Bits reagieren, wie z.B. Audio- oder Videoströme. Allerdings ist in diesem Fall auch eine zusätzliche Lösung zur Resynchronisation nach Synchronisationsfehlern erforderlich.[89] Weiterhin müssen Daten und Übertragungskanal so beschaffen sein, daß die Bedingung Feedbacklänge gleich Blocklänge nicht problematisch wird. Das ist bei der Übertragung hoher Datenmengen auf schnellen Kanälen, wie z.B. Videokonferenzen, der Fall.

3.5.1.4 Sicherheit von symmetrischen Kryptosystemen

Die Sicherheit von symmetrischen Kryptosystemen ist abhängig von der Stärke des Algorithmus selber und der Schlüssellänge. Die Stärke des Algorithmus ist von größter Wichtigkeit, jedoch äußerst schwer nachzuweisen. Ein guter Algorithmus ist gemäß der Terminologie aus Abschnitt 3.3 praktisch sicher, das heißt, es gibt keine bessere Möglichkeit, ein Kryptosystem zu brechen, als sämtliche möglichen Schlüssel in einem Brute-force-Angriff auszuprobieren.[90] Unter der Annahme, daß der vorliegende Algorithmus praktisch sicher ist, ist die Wahl der Schlüssellänge folglich

86 Vgl. ebenda, S. 203 ff.
87 Vgl. ebenda, S. 237.
88 Vgl. ebenda, S. 208.
89 Vgl. ebenda, S. 204.
90 Vgl. ebenda, S. 151.

nach der Wahl des Algorithmus die wichtigste Überlegung zur Sicherheit eines Kryptosystems. Der Schlüssel sollte so lang sein, daß ein Brute-force-Angriff in dem Zeitraum, in dem die chiffrierten Daten geheim bleiben müssen, mit den Ressourcen, die den potentiellen Angreifern zur Verfügung stehen, nicht möglich ist. Unter Berücksichtigung eines schnellen Zuwachses an Rechenleistung werden, abhängig von Geheimhaltungszeitraum und potentiellen Angreifern, Schlüssellängen von 64 Bit (Geheimhaltungszeitraum unter 10 Jahren, potentielle Angreifer sind durchschnittlich stark, z.B. Individuen oder Konkurrenten) bis 128 Bit (Geheimhaltungszeitraum mehrere Jahrzehnte, potentielle Angreifer sind sehr stark, z.B. fremde Regierungen) empfohlen.[91] Tab. 1 des Anhangs führt einige empfohlene Schlüssellängen für Beispieldaten unterschiedlicher Geheimhaltungszeiträume auf.

3.5.2 Asymmetrische Algorithmen (Public-key-Algorithmen)

3.5.2.1 Grundlagen

Das Konzept der *asymmetrischen Algorithmen*, in der Literatur meist als *Public-key-Algorithmen* bezeichnet, wurde 1976 von WHITFIELD DIFFIE und MARTIN HELLMAN vorgestellt.[92] Hier sind die Schlüssel zur Chiffrierung und Dechiffrierung generell voneinander verschieden und lassen sich nicht mit vertretbarem Aufwand voneinander ableiten.[93] Diese Eigenschaft macht das Schlüsselmanagement deutlich leichter, als bei symmetrischen Algorithmen, da einer der beiden Schlüssel gefahrlos öffentlich bekannt gemacht werden kann, ohne die Sicherheit des anderen zu kompromittieren.[94] Der bekannt gemachte Schlüssel wird als *öffentlicher Schlüssel* (englisch: *public key*), der geheimgehaltene Schlüssel als *privater Schlüssel* (englisch: *private key*) bezeichnet.[95]

Die asymmetrischen Kryptosysteme basieren mathematisch auf *Einweg-Funktionen mit Falltür*. *Einweg-Funktionen* sind Funktionen, die nicht umkehrbar sind, die also in eine Richtung problemlos berechenbar sind, jedoch die Berechnung der Umkehrfunktion unmöglich oder rechnerisch undurchführbar ist. Bei bekanntem x ergibt sich y leicht aus $y = f(x)$. Bei bekanntem y jedoch ist x nicht bestimmbar. Eine *Einweg-*

---

91 Vgl. ebenda, S. 152 f. u. 166 f.
92 Vgl. ebenda, S. 31.
93 Vgl. **Buchmann, Johannes**, a.a.O., S. 57.
94 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 5 u. 32.
95 Vgl. **Buchmann, Johannes**, a.a.O., S. 57.

Funktion mit Falltür ist ein spezieller Typ von Einweg-Funktionen, bei dem die Umkehrung rechnerisch undurchführbar ist, es sei denn, man ist im Besitz einer zusätzlichen Information *z*, sozusagen dem Schlüssel zur Falltür. Dann ist die Umkehrung leicht möglich. Im Falle der asymmetrischen Kryptosysteme entspricht der private Schlüssel der zusätzlichen Information, die die Falltür auslöst. [96]

Möchte ein Empfänger chiffrierte Nachrichten empfangen, so muß er seinen Chiffrierungsschlüssel veröffentlichen. Nun kann ihm jeder, der im Besitz dieses öffentlichen Schlüssels ist, Nachrichten senden, aber nur der Empfänger selber kann die Nachrichten dechiffrieren, da nur er im Besitz des Dechiffrierungsschlüssels ist. In einem Netzwerk kann das Schlüsselmanagement für die einzelnen Teilnehmer durch diese Vorgehensweise erleichtert werden, indem z.B. die öffentlichen Schlüssel eines jeden Teilnehmers in einer zentralen Schlüsseldatenbank verwaltet werden.[97] Es muß nicht mehr für jede mögliche Paarung von Kommunikationspartnern ein eigener Schlüssel generiert werden, es genügt, wenn jeder Teilnehmer ein Schlüsselpaar generiert. Die Anzahl der Schlüssel im System nimmt also nur linear mit der Teilnehmerzahl zu.[98] Außerdem ist der Schlüsselaustausch sehr vereinfacht, da ein Dritter, der die Kommunikation abhört, trotz Besitzes des öffentlichen Schlüssels und des Chiffretextes weder den privaten Schlüssel, noch den Klartext ableiten kann.[99]

Leider sind Public-key-Algorithmen mit einigen entscheidenden Nachteilen verbunden, die ihre Verwendbarkeit stark einschränken. Public-key-Algorithmen sind deutlich langsamer als symmetrische Algorithmen, generell mindestens um den Faktor 1.000.[100] Das liegt an der rechnerischen Komplexität, die das Prinzip der Einweg-Funktionen mit sich bringt.[101] Dadurch sind asymmetrische Algorithmen für die Chiffrierung von größeren Datenmengen oder längeren Nachrichten unbrauchbar. Ein anderer Aspekt ist die Sicherheit von Public-key-Algorithmen, die im folgenden Abschnitt genauer erläutert wird. Letztendlich folgt, daß Public-key-Algorithmen nicht als Substitut für symmetrische Algorithmen verwendet werden können, sondern nur für dedizierte Aufgaben in kryptografischen Protokollen geeignet sind, wie z.B.

---

96 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 29 ff.
97 Vgl. ebenda, S. 31 f.
98 Vgl. **Münzenberger, M.:** „Public-Key-Kryptosysteme“, in: HMD – Theorie und Praxis der Wirtschaftsinformatik, 33. Jg. (1996), Heft 190, S. 35.
99 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 31 f.
100 Vgl. ebenda, S. 33 u. 461.
101 Vgl. **Heuser, Ansgar**, a.a.O., S. 11.

den gesicherten Schlüsselaustausch für eine anschließende Kommunikation mittels eines symmetrischen Algorithmus.[102]

3.5.2.2 Sicherheit von asymmetrischen Kryptosystemen

Die wesentlich komplexeren mathematischen Zusammenhänge bei asymmetrischen Algorithmen machen es noch schwieriger, kryptografisch sichere Algorithmen zu entwerfen. Von der Vielzahl an Public-key-Algorithmen, die seit 1976 vorgeschlagen wurden, sind daher viele unsicher. Von denen, die als sicher betrachtet werden können sind wiederum viele unpraktikabel, entweder durch zu große Schlüssel oder eine zu große Nachrichtenexpansion.[103]

Außerdem sind Public-key-Algorithmen prinzipbedingt anfällig für Chosen-plaintext-Angriffe, da es jedem Dritten möglich ist, Nachrichten mit dem öffentlichen Schlüssel zu chiffrieren. Dies ist insbesondere dann von Bedeutung, wenn es relativ wenige mögliche Klartexte gibt. In diesem Falle ist auch schon die Information, daß ein bestimmter abgefangener Chiffretext nicht einem bestimmten Klartext entspricht, von Wert. Dem kann dadurch entgegengewirkt werden, daß dem Klartext eine Reihe von zufälligen Bits bei der Verschlüsselung angehängt werden. In diesem Fall ergibt ein und derselbe Klartext nie zweimal denselben Chiffretext. Viele Public-key-Algorithmen sind jedoch auch empfindlich gegenüber einem Chosen-ciphertext-Angriff.[104]

Ein weiteres Problem liegt in der minimal erforderlichen Schlüssellänge. Die öffentlichen Schlüssel müssen auf irreversible Weise aus den privaten Schlüsseln generiert werden. Das ist allerdings nur sehr schwer möglich, so daß man sich darauf verlassen muß, daß die Umkehrung der Schlüsselgenerierung rechnerisch undurchführbar ist. Die Schlüsselgenerierung stützt sich auf die schwere Lösbarkeit bestimmter mathematischer Probleme, wie z.B. die Faktorisierung extrem großer Zahlen.[105] Hier werden die öffentlichen Schlüssel durch Multiplikation von großen Primzahlen erzeugt. Das Produkt der Primzahlen ist der *Modulus*. Die Dechiffrierungsoperationen sind nur möglich, wenn die Primfaktorzerlegung des Modulus bekannt ist.[106] Problematisch ist jedoch, daß Probleme wie die Faktorisierung großer Zahlen mathematisch

---

[102] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 33 und 461 f.
[103] Vgl. ebenda, S. 461.
[104] Vgl. ebenda, S. 33 u. 461f.
[105] Vgl. ebenda, S. 158 ff.
[106] Vgl. **Leiberich, Otto:** „Kryptographie – Ist alles unsicher?“, in: KES, 15. Jg. (1999), Heft 5, S. 16.

nicht als schwer lösbar nachgewiesen werden können. Tatsächlich tauchen z.B. regelmäßig neue Algorithmen auf, mit denen noch größere Zahlen noch schneller faktorisiert werden können.[107] So wurde beispielsweise 1999 vom Institut für Mathematik und Computer in Amsterdam mit Hilfe von 300 Großrechnern und einem Cray-Supercomputer eine Zahl von 155 Dezimalstellen faktorisiert.[108] Aus diesem Grunde müssen die Schlüssel von Public-key-Kryptosystemen deutlich länger sein, als bei symmetrischen Kryptosystemen. Weiterhin muß damit gerechnet werden, daß die für ausreichende Sicherheit erforderliche Schlüssellänge in Zukunft schnell größer werden wird.[109]

Die Schlüssellängen von Public-key-Kryptosystemen müssen daher äußerst vorsichtig und vorausschauend bestimmt werden, da die Länge des Geheimhaltungszeitraums noch stärker ins Gewicht fällt, als bei symmetrischen Kryptosystemen. Empfohlen werden Schlüssellängen zwischen 1024 und 2048 Bit, abhängig von der Stärke der potentiellen Angreifer und der Länge des Geheimhaltungszeitraums.[110] Detailliertere Informationen finden sich in Tab. 2 des Anhangs.

3.5.3 Hybrid-Verfahren

3.5.3.1 Definition

Als *Hybrid-Verfahren* bezeichnet man eine Kombination aus symmetrischen und asymmetrischen Algorithmen, durch die das einfache Schlüsselmanagement von Public-key-Kryptosystemen mit den Geschwindigkeitsvorteilen von symmetrischen Kryptosystemen verbunden werden kann. Will ein Sender einem Empfänger eine Nachricht senden, so erzeugt er für einen symmetrischen Algorithmus einen *Sitzungsschlüssel* und chiffriert damit den Klartext. Dieser Sitzungsschlüssel wird nur für diese eine Nachricht verwendet. Dann verschlüsselt er den Sitzungsschlüssel mit einem Public-key-Algorithmus und dem dazugehörigen öffentlichen Schlüssel des Empfängers und hängt das Ergebnis an den Chiffretext der Nachricht an. Der Empfänger kann nun mit seinem privaten Schlüssel den Sitzungsschlüssel dechiffrieren und mit diesem wiederum die Nachricht dechiffrieren. Der Sitzungsschlüssel ist kurz

---

[107] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 158 f.
[108] Vgl. **Leiberich, Otto**, a.a.O., S. 16 f.
[109] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 160 ff.
[110] Vgl. ebenda, S. 160 ff.

genug, daß die Geschwindigkeitsnachteile von Public-key-Algorithmen nicht ins Gewicht fallen und ein einfaches Schlüsselmanagement ist möglich.[111]

3.5.3.2 Sicherheitsaspekte von Hybrid-Verfahren

Bei dem Entwurf von Kryptosystemen, die sowohl symmetrische, als auch asymmetrische Algorithmen verwenden ist es wichtig, darauf zu achten, daß beide Bestandteile den gleichen Sicherheitsstandard bieten. Unter der Voraussetzung, daß sichere Algorithmen verwendet werden, ist es notwendig, die Schlüssellängen derart auszuwählen, daß beispielweise die Schwierigkeit der Faktorisierung der Schlüssel des Public-key-Algorithmus in etwa der Schwierigkeit eines Brute-force-Angriffs auf den symmetrischen Algorithmus entspricht.[112] Tab. 3 des Anhangs listet entsprechende Schlüssellängen-Paare auf.

Da generell die öffentlichen Schlüssel länger gültig sind, als die symmetrischen Schlüssel (in der Regel sind diese nur Sitzungsschlüssel) und mit ihnen sehr sensible Informationen, wie z.B. die Sitzungsschlüssel, geschützt werden, sollten die Schlüssellängen für die Public-key-Algorithmen sogar so gewählt werden, daß sie größere Sicherheit bieten, als die symmetrischen Schlüssel.[113]

3.5.4 Hash-Funktionen

3.5.4.1 Grundlagen

Allgemein ausgedrückt ist eine *Hash-Funktion* eine Funktion ($H(M)$), die aus einer Bitfolge beliebiger Länge eine Bitfolge fester Länge – den *Hash-Wert* – errechnet.[114] In der Kryptografie werden Hash-Funktionen meist als eine Art Prüfsumme über eine Nachricht verwendet. Man möchte gewissermaßen einen Fingerabdruck eines Dokuments erzeugen, anhand dessen nachträgliche Veränderungen nachweisbar sind. Dabei macht man sich die Eigenschaft von Hash-Funktionen zunutze, daß schon die Änderung eines einzelnen Bits der Nachricht den Hash-Wert verändert.[115] Dieser Einsatz von Hash-Funktionen verlangt, daß diese Einweg-Charakter haben. Das heißt, daß die Berechnung des Hash-Wertes einfach ist, jedoch die Erzeugung eines

---

[111] Vgl. **Buchmann, Johannes**, a.a.O., S. 111 f.
[112] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 165.
[113] Vgl. ebenda, S. 165.
[114] Vgl. **Oppliger, Rolf**, a.a.O., S. 453.
[115] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 30.

Dokuments, das einem bestimmten Hash-Wert entspricht, unmöglich oder mindestens schwierig ist. Weiterhin sollte eine gute Hash-Funktion *kollisionsresistent* sein, d.h. daß es mindestens schwierig ist, zwei Dokumente zu erzeugen, die denselben Hash-Wert haben (ein solcher Umstand wird als *Kollision* bezeichnet).[116]

Hash-Funktionen finden Anwendung als Prüfsummen zur Integritätsprüfung von Nachrichten und im Bereich der digitalen Signaturen, die im Abschnitt 5 behandelt werden.

3.5.4.2 Sicherheitsaspekte von Hash-Funktionen

Ein böswilliger Kommunikationspartner oder Dritter könnte ein Interesse daran haben, ein Dokument unbemerkt und nicht nachweisbar auf dem Kommunikationsweg oder nachträglich zu verändern. Genau vor dieser Intention sollen aber Hash-Funktionen schützen. Dieser Schutz kann jedoch durch den Angreifer überwunden werden, wenn es ihm gelingt, eine Kollision zu erzeugen, also das Originaldokument derart zu verändern, daß es denselben Hash-Wert ergibt. Ein Angriff auf eine Hash-Funktion versucht genau das.[117]

Hash-Funktionen sind nicht injektiv, denn in aller Regel wird eine Nachricht länger sein, als der zu erzeugende Hash-Wert. Folglich gibt es mehr mögliche Nachrichten, als mögliche Hash-Werte. Es muß somit mehrere Nachrichten geben, die zu demselben Hash-Wert führen.[118] Dieser Umstand wird durch den sogenannten *Geburtstagsangriff* auf Hash-Funktionen ausgenutzt. Der Name leitet sich von dem in der Statistik bekannten *Geburtstagsparadox* ab. Es besagt folgendes: Damit die Wahrscheinlichkeit, in einer Gruppe von Menschen einen zu finden, der einen bestimmten, festgelegten Geburtstag hat, größer als 0,5 ist, sind mindestens 253 Menschen erforderlich. Um mit einer Wahrscheinlichkeit von mehr als 0,5 in einer Gruppe von Menschen zwei zu finden, die denselben (nicht spezifizierten) Geburtstag haben, sind jedoch nur 23 Menschen erforderlich. Die letztere Variante ist folglich deutlich wahrscheinlicher als die erstere. Dieses Paradox trifft auch auf Hash-Funktionen zu, wenn man den Hash-Wert analog zum Geburtstag betrachtet. Es mag also schwierig sein, eine Nachricht zu finden, die einen bestimmten Hash-Wert hat

---

[116] Vgl. **Preneel, Bart:** „The State of Cryptographic Hash Functions“, in: Lectures on Data Security – Modern Cryptology in Theory and Practice, **Hrsg.: Damgård, Ivan B.**, Berlin et al. 1999, S. 159 ff.

[117] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 429 f.

[118] Vgl. **Buchmann, Johannes**, a.a.O., S. 165.

(es müßten bei einem n-Bit Hash-Wert 2^n zufällige Nachrichten erzeugt werden, damit die Trefferwahrscheinlichkeit größer als 0,5 ist). Es ist aber deutlich einfacher, irgendzwei beliebige Nachrichten zu finden, die den gleichen, jedoch unbestimmten, Hash-Wert haben (die Wahrscheinlichkeit, eine Kollision zu finden wird schon bei der Erzeugung von $2^{n/2}$ zufälligen Nachrichten größer als 0,5).[119]

Ein Angreifer kann sich diesen Umstand zunutze machen, wenn er z.B. zwei unterschiedliche Versionen eines Dokuments vorbereitet und an beiden systematisch kosmetische Änderungen vornimmt (Einfügen von Leerstellen, etc.), bis beide denselben Hash-Wert ergeben.[120]

Hash-Funktionen, die 64-Bit-Hash-Werte erzeugen sind nicht sicher genug, da sie mit heutigen Rechenleistungen Geburtstagsangriffen nicht mehr standhalten können. Deshalb verwenden die meisten praktisch eingesetzten Verfahren 128-Bit-Hash-Werte. Auch dieser Wert wird auf lange Sicht hin nicht ausreichen. Moderne Hash-Funktionen haben daher eine Hash-Wert-Länge von 160 Bit.[121]

3.6 Kryptografische Protokolle

3.6.1 Definition

Jede Kommunikation läuft zwangsweise innerhalb eines *Protokolls* ab, gleichgültig ob man sich dessen ausdrücklich bewußt ist, oder nicht. Protokolle sind Vereinbarungen über den organisatorischen Ablauf der Kommunikation.[122] Sie bestehen generell aus einer Reihe von sequentiell durchzuführenden Schritten und involvieren mindestens zwei Teilnehmer. Als *kryptografisches Protokoll* wird ein Protokoll bezeichnet, bei dem kryptografische Algorithmen zum Einsatz kommen.[123]

Demnach ist jede chiffrierte Kommunikation auch zwangsweise in ein kryptografisches Protokoll eingebunden. Im Lichte dessen wird klar, daß die Benutzung eines guten Kryptosystems alleine keine Sicherheit der Kommunikation garantieren kann. Das Kryptosystem muß in ein sicheres kryptografisches Protokoll eingebunden sein, welches die Kommunikation so handhabt, daß sie nicht angreifbar ist.

---

119 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 166.
120 Vgl. ebenda, S. 430.
121 Vgl. ebenda, S. 430.
122 Vgl. **Stahlknecht, Peter; Hasenkamp, Ulrich:** Einführung in die Wirtschaftsinformatik, 8. Aufl., Berlin et al. 1997, S. 124.
123 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 21 f.

Darüber hinaus dienen kryptografische Protokolle dem Zweck, in anonymen Umgebungen wie Computer-Netzwerken Fairness und Vertrauen zwischen einander latent mißtrauenden Teilnehmern herzustellen. Sie sollen also vor Betrug durch unehrliche Teilnehmer schützen, indem sie sicherstellen, daß es keinem Teilnehmer möglich ist, Aktionen durchzuführen oder Informationen zu gewinnen, die nicht im Protokoll so spezifiziert sind. Es darf nicht möglich sein, das Protokoll zu unterwandern.[124]

3.6.2 Angriffe auf kryptografische Protokolle

Angriffe auf kryptografische Protokolle können von Personen innerhalb und außerhalb des Protokolls ausgehen. Versucht ein Außenstehender, durch Observation oder Abhören der Kommunikation Informationen über das Protokoll oder die darin eingebundenen Personen zu gewinnen, so spricht man von einem *passiven Angriff*. Dem gegenüber steht der *aktive Angriff*, bei dem ein Außenstehender versucht, das Protokoll zu seinen Gunsten zu manipulieren. Dies kann auf vielfältige Weise geschehen. Der Angreifer kann beispielsweise eine fremde Identität vortäuschen, gefälschte Nachrichten in das Protokoll einschleusen, bestehende Nachrichten löschen oder gegen gefälschte austauschen, alte Nachrichten wiederholen, Kommunikationskanäle unterbrechen oder gespeicherte Informationen manipulieren. Die Absichten des Angreifers könnten beispielsweise sein, Informationen zu gewinnen, die Systemleistung zu verringern, bestehende Daten zu verändern oder unautorisierten Zugriff auf Daten oder andere Ressourcen zu erlangen. Angriffe, die von Teilnehmern des Protokolls ausgehen werden als *Betrug* bezeichnet. Ein *passiver Betrug* liegt vor, wenn ein Teilnehmer das Protokoll zwar einhält, jedoch versucht, mehr Informationen zu gewinnen, als es das Protokoll vorsieht. Weicht ein Teilnehmer von dem Protokoll ab, stört er das Protokoll oder lügt er im Verlaufe des Protokolls, so spricht man von *aktivem Betrug*.[125]

Passive Angriffe sind schwer zu erkennen. Ein Protokoll sollte so entworfen sein, daß es passive Angriffe verhindert. Aktive Angriffe sind wesentlich schwerwiegender als passive Angriffe. Ein Protokoll sollte deshalb die Möglichkeit bieten, aktive Angriffe zu erkennen und zu unterbinden. Es ist schwierig, die Sicherheit eines Protokolls aufrecht zu erhalten, wenn die meisten Teilnehmer aktive Betrüger sind. Es

---

[124] Vgl. ebenda, S. 22 f.
[125] Vgl. ebenda, S. 27.

sollte jedoch für legitime Teilnehmer möglich sein, aktiven Betrug zu erkennen. Gute Protokolle sollten außerdem passiven Betrug unmöglich machen.[126]

4 Verfahren zur Gewährleistung der Vertraulichkeit

4.1 Symmetrische Algorithmen

4.1.1 Data Encryption Standard (DES)

4.1.1.1 Beschreibung des Algorithmus

Der Data Encryption Standard (DES) wurde von IBM für das damalige National Bureau of Standards (NBS), heute National Institute of Standards and Technology (NIST), entwickelt und 1975 vom NBS veröffentlicht. 1981 wurde DES vom American National Standards Institute (ANSI) als Standard für den privaten Sektor genehmigt. Auch die International Organization for Standardization (ISO) hat DES als Standard genehmigt. DES ist nunmehr seit über 20 Jahren ein weltweiter Standard.[127]

Die Zertifizierung von DES durch das NBS und später das NIST wurde trotz wachsender Bedenken bezüglich der Sicherheit des Algorithmus immer wieder erneuert,[128] zuletzt im Jahre 1999.[129]

Der Algorithmus wurde von der National Security Agency (NSA) evaluiert, als sicher erklärt und für den Export genehmigt. Aus diesem Grunde war die NSA auch in die Entwicklung des Algorithmus involviert und hat einige Spezifikationen des ursprünglichen Vorschlags von IBM geändert.[130]

DES ist eine Blockchiffre mit einer Blockgröße von 64 Bit. Die Schlüssellänge beträgt ebenfalls 64 Bit.[131] Allerdings ist jedes achte Schlüsselbit ein Prüfbit und wird ignoriert. Die effektive Schlüssellänge beträgt somit 56 Bit.[132]

[126] Vgl. ebenda, S. 27.
[127] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 265 ff.
[128] Vgl. ebenda, S. 269 f.
[129] Vgl. **National Institute of Standards and Technology:** FIPS-PUB 46-3, Data Encryption Standard (DES), Online im Internet, http://csrc.nist.gov/publications/fips/fip46-3/fips46-3.pdf, Stand: 25.10.1999, Abruf: 05.01.2001, S. 4.
[130] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 266 f.
[131] Vgl. **National Institute of Standards and Technology**, 1999, a.a.O., S. 8.
[132] Vgl. ebenda, S. 19.

Die Chiffrierung eines 64-Bit-Blocks mit DES beginnt mit einer Permutation der Bits nach einer festen Tabelle. Die darauffolgenden Chiffrierungsoperationen werden iterativ 16 mal wiederholt.[133]

Zunächst wird der Eingabeblock in zwei Hälften zu je 32 Bits aufgeteilt. Diese werden als linke und rechte Hälfte bezeichnet. Danach wird der 56-Bit-Schlüssel in zwei Hälften zu je 28 Bits aufgeteilt und diese beiden Hälften werden, abhängig von der Nummer der Iteration, um 1 oder 2 Bits linksherum rotiert. Danach wird der Schlüssel wieder zusammengefügt und es werden nach einer festen Tabelle 48 Bits aus den 56 ausgewählt. Bei der Auswahl wird auch eine weitere Permutation der Schlüsselbits durchgeführt. Der rotierte Schlüssel wird gespeichert, denn er dient als Ausgangsschlüssel für die nächste Iteration.[134] So wird sichergestellt, daß für jede Iteration eine unterschiedliche Teilmenge der Schlüsselbits verwendet wird.[135]

Nun wird die rechte Hälfte des Eingabeblocks nach einer festen Tabelle auf 48 Bit vergrößert. Dabei werden einige Bits zweimal verwendet und es findet auch eine Permutation statt. Diese 48 Bits werden nun per XOR mit dem soeben gewonnenen 48-Bit-Teilschlüssel verknüpft.[136] Da sich einige Bits des Blocks zweimal auswirken, wird der Zustand, daß jedes Chiffretextbit von jedem Klartextbit und jedem Schlüsselbit abhängig ist, schneller erreicht.[137]

Darauf folgt eine Substitutionsoperation. Dabei werden durch acht fest vorgegebene Tabellen, die sogenannten *S-Boxen*, je 6 Bits des Ergebnisses der XOR-Operation durch 4 Bits ausgetauscht. Dabei verarbeitet die erste S-Box die ersten 6 Bits, die zweite S-Box die darauffolgenden 6 Bits, und so weiter. Dadurch wird auch die Länge der Daten wieder auf 32 Bits reduziert.[138] Diese Substitutionsoperation ist der wichtigste Schritt bei der Verschlüsselung, da die Operation nicht linear und damit schwer zu analysieren ist.[139]

Die 32 Bits aus der S-Box-Substitution werden nun ein weiteres mal einer Permutation nach einer festen Tabelle unterzogen, der sogenannten *P-Box-Permutation*. Das Ergebnis der Permutation wird per XOR mit der bisher unbehandelten linken Hälfte

---

[133] Vgl. ebenda, S. 10 f.
[134] Vgl. ebenda, S. 11 u. 19 ff.
[135] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 273.
[136] Vgl. **National Institute of Standards and Technology**, 1999, a.a.O., S. 11 u. 13.
[137] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 273.
[138] Vgl. **National Institute of Standards and Technology**, 1999, a.a.O., S. 14 f. u. 17 f.
[139] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 275.

des Eingabeblocks verknüpft.[140] Das Ergebnis dieser Operation wird die neue rechte Hälfte des Blocks, die alte rechte Hälfte wird die neue linke Hälfte. Dieser neu zusammengefügte Block ist die Eingabe für die nächste Iteration.[141]

Nach der sechzehnten und letzten Iteration wird anders verfahren. Das Ergebnis des letzten XOR wird auf die linke Seite und die alte rechte Seite (das Ergebnis der fünfzehnten Iteration) auf die rechte Seite des Ergebnisblocks gelegt. Die beiden Hälften sind also im Vergleich zu den vorhergehenden Permutationen vertauscht.[142] Diese Vorgehensweise ist notwendig, damit der beschriebene Algorithmus auch zur Dechiffrierung angewendet werden kann.[143]

Dieser Block wird nun ein letztes Mal einer Permutation unterzogen. Diese ist genau die Inverse zur ersten Permutation vor den Iterationen. Das Ergebnis dieser Permutation ist der fertige, 64 Bit lange Chiffretextblock.[144]

Zur Dechiffrierung eines Chiffretextblocks wird der oben beschriebene Algorithmus genau so erneut durchlaufen. Der einzige Unterschied ist, daß die Funktion zur Schlüsselgenerierung rückwärts laufen muß, das heißt, der Schlüssel der sechzehnten Chiffrierungsiteration muß für die erste Dechiffrierungsiteration verwendet werden.[145] Dies wird dadurch erreicht, daß der sechzehnte Schlüssel generiert wird und ohne Bitrotation für die erste Dechiffrierungsiteration verwendet wird. In den darauffolgenden Iterationen müssen die Bits der beiden Schlüsselhälften dann nur rechtsherum anstatt linksherum rotiert werden.[146]

DES ist für alle vier Modi von Blockchiffren definiert: ECB, CBC, CFB und OFB.[147] Der Algorithmus selber wird dabei nicht verändert, es müssen nur die Ein- und Ausgaben des Algorithmus entsprechend der Vorgaben von Abschnitt 3.5.1.3 umgeleitet und verknüpft werden.[148]

Das NIST hat DES 1999 auch für mehrfache Chiffrierung als Triple-DES definiert. Dabei wird jeder Block dreimal mit dem DES-Algorithmus behandelt, und zwar

---

140 Vgl. **National Institute of Standards and Technology**, 1999, a.a.O., S. 15 u. 18.
141 Vgl. ebenda, S. 11.
142 Vgl. ebenda, S. 11.
143 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 277.
144 Vgl. **National Institute of Standards and Technology**, 1999, a.a.O., S. 10.
145 Vgl. ebenda, S. 12.
146 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 277.
147 Vgl. **National Institute of Standards and Technology**, 1999, a.a.O., S. 4.
148 Vgl. **Biham, Eli:** „On Modes of Operation“, in: Fast Software Encryption – Cambridge Security Workshop Proceedings, **Hrsg.: Anderson, Ross J.**, Berlin et al. 1994, S. 118 f.

einmal im Chiffrierungsmodus, einmal im Dechiffrierungsmodus, und das dritte Mal wieder im Chiffrierungsmodus. Dementsprechend wird zur Dechiffrierung zuerst der Dechiffrierungsmodus, dann der Chiffrierungsmodus und dann wieder der Dechiffrierungsmodus angewandt. Es können dabei für jeden der drei Schritte unterschiedliche Schlüssel verwendet werden oder es können der erste und dritte Schlüssel identisch sein. Sind alle drei Schlüssel identisch, ist Triple-DES äquivalent zum normalen DES. Auch Triple-DES ist in allen vier Modi für Blockchiffren (ECB, CBC, CFB und OFB) definiert.[149]

4.1.1.2 Sicherheit von DES

Die Beteiligung der NSA am Entwurf von DES hat in den Rängen der Kryptografen weltweit zu vielfältigen Gerüchten und Spekulationen geführt. Es heißt, die NSA habe IBM dazu gebracht, die ursprünglich geplante Schlüssellänge von 128 Bit (mit 16 Prüfbits, also effektiv 112 Bit) auf die oben erwähnten 64 Bit (mit 8 Prüfbits, also effektiv 56 Bit) zu reduzieren. Außerdem wird behauptet, die NSA habe die S-Boxen, die IBM vorgeschlagen hatte, verändert.[150] Es herrscht Skepsis darüber, ob dies geschah, um Schwächen des Algorithmus auszumerzen, oder ob sich die NSA eine Hintertür in DES einbauen wollte, so daß es ihr leicht möglich ist, DES-chiffrierte Daten zu dechiffrieren. Aus diesen Gründen wurde DES über die Jahrzehnte hinweg äußerst gründlich analysiert und den ausgefeiltsten kryptoanalytischen Angriffen ausgesetzt, um mögliche Schwächen aufzudecken.[151] Die Ergebnisse dieses Prozesses werden im folgenden kurz zusammengefaßt.

Der Algorithmus zur Generierung der Iterationsschlüssel führt dazu, daß bestimmte Schlüssel schwächere Sicherheit bieten, als andere. DES hat vier sogenannte *schwache Schlüssel*. Das sind Schlüssel, bei denen der erzeugte Iterationsschlüssel bei allen 16 Iterationen identisch ist. Das ist der Fall, wenn die beiden Hälften der 56 effektiven Schlüsselbits jeweils ausschließlich aus 0-Bits oder 1-Bits bestehen. Weiterhin gibt es zwölf sogenannte *semischwache Schlüssel*. Semischwache Schlüssel erzeugen anstatt sechzehn nur zwei unterschiedliche Iterationsschlüssel, die jeweils acht mal verwendet werden. Das führt dazu, daß ein semischwacher Schlüssel einen korrespondierenden, ebenfalls semischwachen Schlüssel besitzt, bei dem diese beiden Ite-

---

[149] Vgl. **National Institute of Standards and Technology**, 1999, a.a.O., S. 16 u. 22.
[150] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 266 f., 278 ff u. 283 f.
[151] Vgl. **Coppersmith, Don:** „The Data Encryption Standard (DES) and ist strength against attacks", in: IBM Journal of Research and Development, 38. Jg. (1994), Heft 3, S. 244.

rationsschlüssel genau in umgekehrter Reihenfolge erzeugt werden. Die Chiffrierung mit einem semischwachen Schlüssel ist folglich identisch mit der Dechiffrierung mit seinem Partner und umgekehrt. Die semischwachen Schlüssel können also in sechs Paare gruppiert werden.[152] Darüber hinaus gibt es 48 *möglicherweise schwache Schlüssel*. Diese Schlüssel erzeugen nur vier verschiedene Iterationsschlüssel, die jeweils vier mal verwendet werden. Das führt zu insgesamt 64 Schlüsseln aus den 2^{56} möglichen Schlüsseln, die nicht verwendet werden sollen, da sie die Sicherheit des Verfahrens vermindern können.[153]

Eine wichtige Frage war, ob DES eine sogenannte *geschlossene Chiffre* ist. Wäre dies der Fall, so wären Mehrfachchiffrierungen mit DES zwecklos, da es für jede Mehrfachchiffrierung eine korrespondierende einfache Chiffrierung gäbe, die das selbe Ergebnis hätte. Symbolisch ausgedrückt gäbe es für jedes Schlüsselpaar K_1 und K_2 einen korrespondierenden Schlüssel K_3, so daß

$$E_{K_2}(E_{K_1}(M)) = E_{K_3}(M)$$

gälte. Darüber hinaus würde diese Eigenschaft bedeuten, daß DES gegenüber einem bestimmten Known-plaintext-Angriff empfindlich wäre, der das Kryptosystem in durchschnittlich 2^{28} Schritten brechen könnte, und damit deutlich effizienter wäre, als ein Brute-force-Angriff. Erst 1992 konnte eindeutig nachgewiesen werden, daß DES nicht geschlossen ist, seine Sicherheit also hierdurch nicht beeinträchtigt werden kann.[154]

Eine Frage bezüglich des Designs von DES betraf die verwendeten S-Boxen und deren Eigenschaften. Man hatte gewisse auffällige Strukturen in den S-Boxen gefunden und fragte sich, warum gerade diese S-Boxen gewählt wurden, und ob dies geschah, um DES gegen Angriffe zu stärken, oder ob die NSA sich eine Hintertür in den Algorithmus eingebaut hatte.[155] Erst als 1990 die israelischen Mathematiker ELI BIHAM und ADI SHAMIR das von ihnen entdeckte Konzept der *differentiellen Kryptoanalyse* veröffentlichten, konnte diese offene Frage geklärt werden. In der differen-

---

[152] Vgl. **Moore, Judy H.; Simmons, Gustavus J.:** „Cycle Structure of the DES with Weak and Semi-Weak Keys“, in: Advances in Cryptology – CRYPTO '86 Proceedings, **Hrsg.: Odlyzko, A. M.**, Berlin et al. 1987, S. 12 f.

[153] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 281 f.

[154] Vgl. **Campbell, Keith W.; Wiener, Michael J.:** „DES is not a Group“, in: Advances in Cryptology – CRYPTO '92 Proceedings, **Hrsg.: Brickell, Ernest F.**, Berlin et al. 1993, S. 512 u. 516.

[155] Vgl. **Brickell, Ernest. F.; Moore, Judy H.; Purtill, M. R.:** „Structure in the S-Boxes of the DES“, in: Advances in Cryptology – CRYPTO '86 Proceedings, **Hrsg.: Odlyzko, A. M.**, Berlin et al. 1987. S. 4 f.

tiellen Kryptoanalyse werden Klartextpaare mit einem bestimmten, festen Unterschied mit dem zu ermittelnden Schlüssel chiffriert (es handelt sich folglich um einen Chosen-plaintext-Angriff) und die Entwicklung der Chiffrierung durch die einzelnen Iterationen verfolgt. Basierend auf den resultierenden Unterschieden in den Chiffretexten können bestimmten Schlüsseln Wahrscheinlichkeiten zugeordnet werden. Werden viele Chiffretextpaare analysiert, so wird irgendwann der gesuchte Schlüssel zutage treten.[156] Dieser Angriff kann in einen Known-plaintext-Angriff umgewandelt werden, der dann aber deutlich mehr Klartexte erfordert.[157] Er funktioniert bei Blockchiffren, die – wie DES – konstante S-Boxen haben.[158] Abhängig von den Eigenschaften der S-Boxen, der Anzahl der Iterationen und anderen Designmerkmalen, kann ein solcher Angriff effizienter sein, als ein Brute-force-Angriff.[159] Interessant ist, daß bei einem vollen DES mit 16 Iterationen ein Angriff mit den Methoden der differentiellen Kryptoanalyse gerade geringfügig weniger effizient ist, als ein Brute-force-Angriff,[160] so daß DES gerade noch als sicher vor differentieller Kryptoanalyse betrachtet werden kann. Erst jetzt veröffentlichte IBM die wirklichen Designkriterien, die hinter den S-Boxen von DES steckten. Der NSA war das Konzept der differentiellen Kryptoanalyse schon in den siebziger Jahren bekannt, und deshalb wurden die S-Boxen optimiert, um der differentiellen Kryptoanalyse widerstehen zu können. Da dieses Angriffskonzept jedoch der wissenschaftlichen Öffentlichkeit noch nicht bekannt war, wurden damals die wirklichen Designkriterien der S-Boxen nicht veröffentlicht, da sonst die differentielle Kryptoanalyse bekannt geworden wäre und die USA einen wertvollen Wissensvorteil hätten preisgeben müssen.[161] Das erklärte auch, warum DES mit 16 Iterationen definiert wurde, obwohl zur Absicherung gegen herkömmliche Angriffsmethoden auch weniger genügt hätten.[162]

Ein ähnliches Angriffskonzept von ELI BIHAM versucht, das Kryptosystem durch die Untersuchung der Beziehungen zwischen unbekannten Schlüsseln zu brechen. Auch dieser Angriff ist gegen DES nicht einsetzbar, denn die unterschiedliche Rotation der

[156] Vgl. **Biham, Eli; Shamir, Adi:** „Differential Cryptanalysis of DES-like Cryptosystems", in: Advances in Cryptology – CRYPTO '90 Proceedings, **Hrsg.: Menenzes, Alfred J.; Vanstone, Scott A.**, Berlin et al. 1991, S. 6 ff.
[157] Vgl. **Biham, Eli; Shamir, Adi:** „Differential Cryptanalysis of the Full 16-round DES", in: Advances in Cryptology – CRYPTO '92 Proceedings, **Hrsg.: Brickell, Ernest J.**, Berlin et al. 1993, S. 495.
[158] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 289.
[159] Vgl. **Biham, Eli; Shamir, Adi**, 1991, a.a.O., S. 17 ff.
[160] Vgl. **Biham, Eli; Shamir, Adi**, 1993, a.a.O., S. 496.
[161] Vgl. **Coppersmith, Don**, a.a.O., S. 244 u. 246 ff.
[162] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 290.

Schlüsselbits abhängig von der Nummer der Iteration zerstört die Grundlagen für den Angriff.[163]

Ein noch neueres Konzept ist das der *linearen Kryptoanalyse*, vorgestellt von MITSURU MATSUI. Hier wird ein Teil der Schlüsselbits durch Ausnutzung von Ungleichgewichten in der Korrelation von Ein- und Ausgabewerten der S-Boxen mit einzelnen Schlüsselbits gewonnen. Das geschieht durch lineare Approximation der S-Boxen. Die restlichen Schlüsselbits werden durch vollständige Suche im verbleibenden Schlüsselraum ermittelt. Diese Angriffsform ist tatsächlich besser, als ein Brute-force-Angriff.[164] In einem von MATSUI durchgeführten Experiment konnte ein DES-Schlüssel innerhalb von 50 Tagen ermittelt werden. Dabei wurden 2^{43} Klartext-Chiffretext-Paare benötigt.[165] TAKESHI SHIMOYAMA und TOSHINOBU KANEKO haben 1998 gezeigt, daß durch mehrfache lineare Approximation in Verbindung mit nichtlinearer Approximation die Anzahl der notwendigen Klartext-Chiffretext-Paare noch einmal deutlich gesenkt werden kann, nämlich auf 73,5 % der von MATSUI benötogten Anzahl von 2^{43}. In einem Experiment gelang die Ermittlung eines DES-Schlüssels innerhalb von 7 Tagen.[166] Auch die lineare Kryptoanalyse hängt stark von der Struktur der S-Boxen ab. Die Designkriterien für die S-Boxen von DES haben diese Angriffsform nicht berücksichtigt, so daß die S-Boxen nicht im Hinblick darauf optimiert werden konnten und DES folglich empfindlich gegenüber linearer Kryptoanalyse ist.[167]

Das größte Problem mit DES ist heute aber der vergleichsweise kleine Schlüsselraum. Mit heutigen Rechenkapazitäten ist eine effektive Schlüssellänge von 56 Bit nicht mehr ausreichend, um vor Brute-force-Angriffen zu schützen. Bereits 1993 hat MICHAEL J. WIENER einen Spezialrechner entworfen, der einen Brute-Force-Angriff auf DES in durchschnittlich 3,5 Stunden mit Erfolgsgarantie nach 7 Stunden durchführen könnte und schon damals für rund 1 Million US$ hätte gebaut werden können. Es wäre naiv, zu glauben, daß nicht inzwischen irgend jemand, sei es eine Ge-

---

163 Vgl. ebenda, S. 290.
164 Vgl. **Matsui, Mitsuru:** „Linear Cryptanalysis Method for DES Cipher", in: Advances in Cryptology – EUROCRYPT '93 Proceedings, **Hrsg.: Helleseth, Tor**, Berlin et al. 1994, S. 388 ff. u. 393 f. [1994 a]
165 Vgl. **Matsui, Mitsuru:** „The First Experimental Cryptanalysis of the Data Encryption Standard", in: Advances in Cryptology – CRYPTO '94 Proceedings, **Hrsg.: Desmedt, Yvo G.**, Berlin et al. 1994, S. 9 f. [1994 b]
166 Vgl. **Shimoyama, Takeshi; Kaneko, Toshinobu:** „Quadratic Relation of S-box and Its Application to the Linear Attack of Full Round DES", in: Advances in Cryptology – CRYPTO '98 Proceedings, **Hrsg.: Krawczyk, Hugo**, Berlin et al. 1998, S. 201 u. 207.
167 Vgl. **Coppersmith, Don**, a.a.O., S. 250.

heimdienstorganisation oder ein Großkonzern, eine solche Maschine gebaut haben könnte.[168] Setzt sich der momentane Trend zu immer größeren Rechenleistungen in immer kürzerer Zeit weiter fort, so werden bald noch schnellere Resultate möglich sein. Daher kann DES in seiner einfachen Form für den ernsthaften Einsatz heute nicht mehr empfohlen werden.[169]

Dem soll der im vorhergehenden Abschnitt erwähnte Triple-DES abhelfen. Dabei bietet die Version mit drei unabhängigen Schlüsseln die größte Sicherheit, hier steigt der Aufwand für einen Brute-force-Angriff von 2^{56} auf 2^{112}. [170] Der Exponent verdoppelt sich nur, da die einzelnen Schlüssel bei der Chiffrierung untereinander nicht in Beziehung gebracht werden. Die Triple-DES-Variante mit zwei unabhängigen Schlüsseln ist deutlich unsicherer, als die Variante mit drei Schlüsseln.[171]

Zusammenfassend ist festzustellen, daß das Design von DES mittlerweile in die Jahre gekommen ist. Die konstanten S-Boxen sind problematisch, besonders, wenn man in Betracht zieht, daß die lineare Kryptoanalyse relativ jung ist und trotzdem schon Fortschritte gemacht hat. Die geringe effektive Schlüssellänge läßt den einfachen DES aus der Betrachtung völlig herausfallen. Triple-DES behebt dieses Problem, jedoch basiert er immer noch auf demselben, über 25 Jahre alten Design. Darüber hinaus ist anzumerken, daß die Mehrfachverschlüsselung aufgrund der fehlenden Beziehungen zwischen den Teilschlüsseln niemals die gleiche Sicherheit bieten kann, wie eine mit der entsprechenden Schlüssellänge entworfene einfache Chiffrierung[172]. Rechenleistung und Kryptoanalyse machen Fortschritte, und so ist die Nutzung eines so alten und derart intensiv untersuchten und attackierten Algorithmus bedenklich.[173]

---

[168] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 153, 283 f. u. 300.
[169] Vgl. **Leiberich, Otto**, a.a.O., S. 16.
[170] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 294 u. 357 ff.
[171] Vgl. **Merkle, Ralph C.; Hellman, Martin E.:** „On the Security of Multiple Encryption“, in: Communications of the ACM, 24. Jg. (1981), Heft 7, S. 466 f.
[172] Vgl. ebenda, S. 467.
[173] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 300 f.

4.1.2 International Data Encryption Algorithm (IDEA)

4.1.2.1 Beschreibung des Algorithmus

XUEJIA LAI und JAMES L. MASSEY stellten 1990 einen Algorithmus vor, den sie Proposed Encryption Standard (PES) nannten.[174] Nachdem BIHAM und SHAMIR ihre Methode der differentiellen Kryptoanalyse vorgestellt hatten, analysierten LAI und MASSEY PES im Hinblick darauf und stellten 1991 eine modifizierte Version vor, die größere Sicherheit vor differentieller Kryptoanalyse bot. Sie nannten den neuen Algorithmus zunächst Improved Proposed Encryption Standard (IPES),[175] änderten 1992 aber den Namen in International Data Encryption Algorithm (IDEA).[176]

IDEA ist eine Blockchiffre mit einer Blockgröße von 64 Bit und einer Schlüssellänge von 128 Bit. IDEA erreicht Konfusion und Diffusion durch ein Konzept der Mischung von Operationen aus unterschiedlichen algebraischen Gruppen. Dabei werden ausschließlich die drei Operationen

- XOR von 16-Bit-Teilblöcken,
- Addition von 16-Bit-Teilblöcken modulo 2^{16} und
- Multiplikation von 16-Bit-Teilblöcken modulo $2^{16} + 1$

verwendet. Dabei werden die Teilblöcke jeweils als Integer-Werte interpretiert.[177]

IDEA bearbeitet den Klartextblock B in vier 16-Bit Unterblöcken $B_{i,j}$ iterativ. i bezeichnet den Teilblock, j die Nummer der Iteration. Dabei werden acht Iterationen durchgeführt und 52 16-Bit-Teilschlüssel K_k benötigt.[178]

Zur Generierung der Teilschlüssel wird zunächst der Originalschlüssel in acht Teile zu je 16-Bit aufgeteilt. Dieses sind bereits die Schlüssel K_1 bis K_8. Dann wird der Schlüssel um 25 Bits linksherum rotiert und das Ergebnis wieder in acht Teile aufgeteilt, diese sind dann K_9 bis K_{16}. Der rotierte Schlüssel wird erneut um 25 Bits links-

---

[174] Vgl. **Lai, Xuejia; Massey, James L.:** „A Proposal for a New Block Encryption Standard“, in: Advances in Cryptology – EUROCRYPT '90 Proceedings, **Hrsg.: Dåmgard, Ivan B.**, Berlin et al. 1991, S. 389. [1991 a]
[175] Vgl. **Lai, Xuejia; Massey, James L.:** „Markov Ciphers and Differential Cryptanalysis“, in: Advances in Cryptology – EUROCRYPT '91 Proceedings, **Hrsg.: Davies, Donald W.**, Berlin et al. 1991, S. 17 f. [1991 b]
[176] Vgl. **Meier, Willi:** „On the Security of the IDEA Block Cipher“, in: Advances in Cryptology – EUROCRYPT '93 Proceedings, **Hrsg.: Helleseth, Tor**, Berlin et al. 1994, S. 371.
[177] Vgl. **Lai, Xuejia; Massey, James L.**, 1991 a, a.a.O., S. 389 f.
[178] Vgl. ebenda, S. 390.

herum rotiert und aufgeteilt, so daß man K_{17} bis K_{24} erhält. Dieser Vorgang wird fortgesetzt, bis K_{52} erreicht ist.[179]

Der Klartextblock wird in die vier Teilblöcke $B_{1,0}$ bis $B_{4,0}$ aufgeteilt, diese sind die Eingabe für die erste Iteration.[180]

Die Iterationen j laufen jeweils wie folgt ab:

(1) Multipliziere $B_{1,j-1}$ mit $K_{1+6(j-1)}$ modulo $2^{16} + 1$
(2) Addiere $B_{2,j-1}$ mit $K_{2+6(j-1)}$ modulo 2^{16}
(3) Addiere $B_{3,j-1}$ mit $K_{3+6(j-1)}$ modulo 2^{16}
(4) Multipliziere $B_{4,j-1}$ mit $K_{4+6(j-1)}$ modulo $2^{16} + 1$
(5) XOR der Ergebnisse der Schritte (1) und (3)
(6) XOR der Ergebnisse der Schritte (2) und (4)
(7) Multipliziere das Ergebnis von Schritt (5) mit $K_{5+6(j-1)}$ modulo $2^{16} + 1$
(8) Addiere die Ergebnisse der Schritte (6) und (7) modulo 2^{16}
(9) Multipliziere das Ergebnis von Schritt (8) mit $K_{6+6(j-1)}$ modulo $2^{16} + 1$
(10) Addiere die Ergebnisse der Schritte (7) und (9) modulo 2^{16}
(11) XOR der Ergebnisse der Schritte (1) und (9)
(12) XOR der Ergebnisse der Schritte (3) und (9)
(13) XOR der Ergebnisse der Schritte (2) und (10)
(14) XOR der Ergebnisse der Schritte (4) und (10)
(15) Vertausche die Ergebnisse der Schritte (12) und (13), so daß gilt:
$B_{1,j} = (11)$, $B_{2,j} = (13)$, $B_{3,j} = (12)$, $B_{4,j} = (14)$

Diese vier Blöcke sind die Eingabe der nächsten Iteration. In der achten und letzten Iteration wird der Tausch der mittleren Teilblöcke (Schritt (15)) ausgelassen. Nach der achten Iteration werden die Teilblöcke einer letzten Transformation unterzogen:

(1) Multipliziere $B_{1,8}$ und K_{49} modulo $2^{16} + 1$
(2) Addiere $B_{2,8}$ und K_{50} modulo 2^{16}
(3) Addiere $B_{3,8}$ und K_{51} modulo 2^{16}
(4) Multipliziere $B_{4,8}$ und K_{52} modulo $2^{16} + 1$

Die Ergebnisse dieser vier Schritte werden in dieser Reihenfolge zusammengefügt und ergeben so den Chiffretextblock.[181]

---

[179] Vgl. ebenda, S. 392 f.
[180] Vgl. ebenda, S. 390.

Die Dechiffrierung läuft prinzipiell identisch ab. Es müssen nur die Teilschlüssel in entsprechend geänderter Reihenfolge verwendet werden und bei Additions- und Multiplikationsoperationen die jeweilige additive bzw. multiplikative Inverse (jeweils modulo 2^{16} bzw. $2^{16} + 1$) des betreffenden Teilschlüssels verwendet werden.[182]

IDEA kann in allen vier Modi von Blockchiffren (ECB, CBC, CFB, OFB) implementiert werden, und auch Mehrfachchiffrierungen sind möglich.[183]

Implementationen von IDEA laufen etwa doppelt so schnell, wie Implementationen von DES.[184]

IDEA ist in Europa und den USA von der schweizer Ascom-Tech AG patentiert. Der Algorithmus ist für nichtkommerzielle Nutzer gebührenfrei, für kommerzielle Nutzung entfällt jedoch eine Lizenzgebühr.[185]

4.1.2.2 Sicherheit von IDEA

Seit seiner Veröffentlichung wurde IDEA von diversen Kryptografen und Kryptoanalytikern untersucht, um mögliche Schwächen aufzudecken. Dabei sind bislang folgende Ergebnisse erzielt worden:

JOAN DAEMEN, RENÉ GOVAERTS und JOOS VANDEWALLE fanden einige größere Klassen von schwachen Schlüsseln für IDEA, die die Eigenschaft haben, daß ein Angreifer deren Verwendung durch einen Chosen-plaintext-Angriff mit nur zwei entsprechend ausgewählten Klartexten leicht erkennen kann. Bei zufälliger Auswahl von Schlüsselwerten ist die Wahrscheinlichkeit, einen solchen schwachen Schlüssel zu erhalten allerdings äußerst gering, nämlich nur 2^{-96}. Abhilfe ist jedoch einfach zu schaffen. Wird beispielsweise jeder der 52 16-Bit-Teilschlüssel K_k vor der Chiffrierung mit dem Hexadezimalwert $0DAE_{16}$ per XOR verknüpft[186], so treten keine schwachen Schlüssel mehr auf.[187]

---

[181] Vgl. **Lai, Xuejia; Massey, James L.**, 1991 b, a.a.O., S. 29.
[182] Vgl. **Lai, Xuejia; Massey, James L.**, 1991 a, a.a.O., S. 392 in Verbindung mit **Lai, Xuejia; Massey, James L.**, 1991 b, a.a.O., S. 28.
[183] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 323 f.
[184] Vgl. ebenda, S. 322.
[185] Vgl. ebenda, S. 325.
[186] $0DAE_{16}$ entspricht 0000110110101110_2. Durch ein XOR mit diesem Wert wird jedes Bit des anderen Operanden invertiert, an dessen Stelle $0DAE_{16}$ eine 1 Aufweist.
[187] Vgl. **Daemen, Joan; Govaerts, René; Vandewalle, Joos:** „Weak Keys for IDEA", in: Advances in Cryptology – CRYPTO '93 Proceedings, **Hrsg.: Stinson, Douglas R.**, Berlin et al. 1994, S. 229 ff.

WILLI MEIER untersuchte die arithmetischen Eigenschaften der Operationen der Iterationsfunktion von IDEA und war in der Lage zu zeigen, daß ein reduzierter IDEA mit nur zwei Iterationen so schneller gebrochen werden kann, als durch einen Bruteforce-Angriff. Allerdings gilt dies bei einem IDEA mit drei Iterationen schon nicht mehr, ein voller IDEA mit acht Iterationen ist dieser Angriffsmöglichkeit gegenüber völlig unempfindlich.[188]

Die differentielle Kryptoanalyse von PES und der modifizierten Version IPES, die nun als IDEA bezeichnet wird, legt nahe, daß differentielle Kryptoanalyse gegen IDEA wahrscheinlich wirkungslos ist.[189] ELI BIHAM ist der Ansicht, daß sein Ansatz der Kryptoanalyse über verwandte Schlüssel ebenfalls bei IDEA nicht funktioniert.[190]

PHILIP HAWKES und LUKE O'CONNOR konnten 1996 nachweisen, daß bei einem modifizierten IDEA mit unabhängigen, gleichverteilten Teilschlüsseln lineare Kryptoanalyse nur bei einem Anteil von $2^{-100,8}$ des gesamten Schlüsselraums durchführbar ist.[191]

JOHN KELSEY, BRUCE SCHNEIER und DAVID WAGNER fanden zwei theoretisch mögliche Angriffe durch Kombination von differentieller Kryptoanalyse und BIHAMS Ansatz über verwandte Schlüssel. Einer dieser Angriffe funktioniert bei einem reduzierten IDEA mit nur drei Iterationen, nicht jedoch bei mehr Iterationen. Der andere Angriff nutzt die Abhängigkeit der Dauer der Chiffrierung vom Aussehen des Schlüssels aus. Er funktioniert bei einem vollen IDEA mit acht Iterationen, wenn man in der Lage ist, die benötigte Rechenzeit pro Aggregat von 2^{20} Chiffrierungen sekundengenau zu messen.[192]

Eine neue Angriffsform, die 1999 von ELI BIHAM, ALEX BIRYUKOV und ADI SHAMIR auf IDEA angewandt wurde, nutzt auf komplizierte Weise sogenannte unmögliche Ereignisse aus. Es wird hier versucht, zwei Ereignisse mit der Wahrscheinlichkeit 1 zu finden, die sich jedoch gegenseitig ausschließen. So können alle Schlüssel als

---

[188] Vgl. **Meier, Willi**, a.a.O., S. 382 ff.
[189] Vgl. **Lai, Xuejia; Massey, James L.**, 1991 b, a.a.O., S. 29 ff. u. 38.
[190] Vgl. **Biham, Eli:** Persönliche Kommunikation mit Bruce Schneier, zitiert bei: **Schneier, Bruce**, 1996, a.a.O., S. 323.
[191] Vgl. **Hawkes, Philip; O'Connor, Luke:** „On Applying Linear Cryptanalysis to IDEA“, in: Advances in Cryptology – ASIACRYPT '96 Proceedings, **Hrsg.: Kim, Kwangjo; Matsumoto, Tsutomu**, Berlin et al. 1996, S. 106 u. 112 ff.
[192] Vgl. **Kelsey, John; Schneier, Bruce; Wagner, David:** „Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER and Triple-DES“, in: Advances in Cryptology – CRYPTO '96 Proceedings, **Hrsg.: Koblitz, Neal**, Berlin et al. 1996, S. 244 ff.

unmöglich ausgeschlossen werden, die zum gleichzeitigen Auftreten dieser Ereignisse führen. Es kann damit systematisch der korrekte Schlüssel ermittelt werden.[193] Diese Angriffsform ist die beste momentan bekannte Angriffsmöglichkeit auf IDEA und ist für reduzierte Varianten mit maximal 4,5 Iterationen effizienter, als ein Brute-force-Angriff, für mehr Iterationen ist sie jedoch ungeeignet.[194]

Noch ist unklar, ob IDEA eine geschlossene Chiffre ist. LAI selber geht nicht davon aus.[195]

Die Schlüssellänge von IDEA ist mit 128 Bit mehr als doppelt so groß, wie bei DES. Das führt zu einem Schlüsselraum von 2^{128} möglichen Schlüsseln, was einen praktischen Brute-force-Angriff wahrscheinlich für alle Zeiten außer Reichweite bleiben läßt.[196]

Eine abschließende Beurteilung der Sicherheit von IDEA ist schwierig, da die vielversprechendsten Angriffsmöglichkeiten erst in jüngster Zeit entdeckt wurden und abzuwarten bleibt, welche Fortschritte hier gemacht werden. Es ist äußerst wahrscheinlich, daß ein IDEA mit vollen acht Iterationen gegen die meisten oben genannten Angriffe immun ist. Diejenigen Möglichkeiten, die auch gegen einen vollen IDEA einsetzbar sind, sind allerdings sehr theoretisch und, obwohl effizienter als ein Brute-force-Angriff, trotzdem wahrscheinlich rechnerisch undurchführbar. Die einzige wirklich bedeutsame Schwachstelle liegt in den schwachen Schlüsseln. Diese treten jedoch äußerst selten auf und ihre Auswirkungen können durch eine minimale Modifikation der Teilschlüsselgenerierung verhindert werden, so daß sie praktisch kein ernsthaftes Problem darstellen dürften. In jedem Falle ist die Sicherheit von IDEA deutlich höher einzuschätzen, als bei DES.

IDEA kann folglich als praktisch sicher gelten, sofern gewährleistet ist, daß

- der Algorithmus mit der vollen Iterationszahl von acht implementiert wird, und
- Maßnahmen gegen die schwachen Schlüssel getroffen werden.

---

[193] Vgl. **Biham, Eli; Biryukov, Alex; Shamir, Adi:** „Miss in the Middle Attacks on IDEA and Khufu", in: Fast Software Encryption – 6th International Workshop Proceedings, **Hrsg.: Knudsen, Lars**, Berlin et al. 1999, S. 124 f.
[194] Vgl. ebenda, S. 126 ff.
[195] Vgl. **Lai, Xuejia:** Persönliche Kommunikation mit Bruce Schneier, zitiert bei: **Schneier, Bruce**, 1996, a.a.O., S. 323.
[196] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 323.

4.1.3 Blowfish

4.1.3.1 Beschreibung des Algorithmus

Blowfish wurde 1993 von BRUCE SCHNEIER vorgestellt. Es handelt sich dabei um eine Blockchiffre mit einer Blockgröße von 64 Bit. Eine Besonderheit von Blowfish ist, daß seine Schlüssellänge variabel ist bis maximal 448 Bit. Blowfish arbeitet iterativ mit 16 Iterationen.[197]

Blowfish benutzt anstatt von Teilschlüsseln im Sinne der zuvor besprochenen Algorithmen ein sogenanntes *P-Array* mit 18 Einträgen (P_1 bis P_{18}) zu je 32 Bit, sowie vier S-Boxen. Diese S-Boxen ordnen 8-Bit-Werten 32-Bit-Werte zu, und sind deshalb Tabellen mit je 256 Einträgen zu je 32 Bit:

$S_{1,0}$ bis $S_{1,255}$,
$S_{2,0}$ bis $S_{2,255}$,
$S_{3,0}$ bis $S_{3,255}$ und
$S_{4,0}$ bis $S_{4,255}$.

Das P-Array und die S-Boxen sind Schlüsselabhängig, sie müssen also vor der Chiffrierung bzw. Dechiffrierung erst aus dem Schlüssel generiert werden.[198] Da bei diesem Vorgang der Blowfish-Algorithmus selber auch zum Einsatz kommt, wird diese sogenannte *Schlüsselexpansion* erst nach dem eigentlichen Algorithmus erläutert. Es wird somit zunächst angenommen, die Schlüsselexpansion sei bereits geschehen, das P-Array und die S-Boxen seien also schon generiert worden.

Zur Chiffrierung wird zunächst der Klartextblock B in zwei Hälften $B_{L,0}$ und $B_{R,0}$ zu je 32 Bit aufgeteilt. Diese beiden Hälften sind die Eingabe für die erste Iteration. jede Iteration i läuft folgendermaßen ab:[199]

(1) XOR von $B_{L,i-1}$ mit P_i
(2) Teile das Ergebnis von Schritt (1) auf in vier 8-Bit-Anteile a, b, c und d
(3) Addiere $S_{1,a}$ und $S_{2,b}$ mod 2^{32}
(4) XOR des Ergebnisses von Schritt (3) mit $S_{3,c}$
(5) Addiere das Ergebnis von Schritt (4) und $S_{4,d}$ mod 2^{32}

---

[197] Vgl. **Schneier, Bruce:** „Description of a New Variable-Lenght Key, 64-Bit Block Cipher (Blowfish)“, in: Fast Software Encryption – Cambridge Security Workshop Proceedings, **Hrsg.: Anderson, Ross J.**, Berlin et al. 1994, S. 195.
[198] Vgl. ebenda, S. 195.
[199] Vgl. ebenda, S. 195 f.

(6) XOR des Ergebnisses von Schritt (5) und $B_{R,i-1}$

(7) Setze $B_{L,i}$ = (6) und $B_{R,i}$ = (1)

Nach der sechzehnten und letzten Iteration wird eine abschließende Transformation durchgeführt:[200]

(1) XOR von $B_{L,16}$ und P_{17}

(2) XOR von $B_{R,16}$ und P_{18}

(3) Setze B_L = (2) und B_R = (1)

(4) Kombiniere B_L und B_R zum fertigen Chiffretextblock

Die Dechiffrierung funktioniert nach dem gleichen Schema, mit dem Unterschied, daß das P-Array in umgekehrter Reihenfolge verwendet wird.[201]

Das P-Array und die S-Boxen werden nach folgendem Schema aus dem Schlüssel K erzeugt:[202]

(1) Initialisiere zunächst das P-Array und dann die vier S-Boxen in Reihenfolge mit einer Zeichenfolge, bestehend aus den Nachkommastellen von π in Hexadezimaldarstellung.

(2) XOR von P_1 mit den ersten 32 Bits von K, P_2 mit den nächsten 32 Bits, etc. bis P_{18}. Gelangt man am Ende des Schlüssels an, so werden wieder die ersten 32 Bits verwendet.

(3) Chiffriere einen Block mit ausschließlich Nullbits mit dem Blowfish-Algorithmus unter Verwendung der soweit initialisierten Tabellen.

(4) Ersetze P_1 und P_2 mit der Ausgabe des Blowfish-Algorithmus aus Schritt (3).

(5) Chiffriere die Ausgabe des Blowfish-Algorithmus aus Schritt (3) erneut mit dem Blowfish-Algorithmus unter Verwendung der modifizierten Tabellen.

(6) Ersetze P_3 und P_4 mit der Ausgabe des Blowfish-Algorithmus aus Schritt (5).

(7) Setze den Prozeß fort, bis das gesamte P-Array und alle vier S-Boxen der Reihe nach mit den sich kontinuierlich verändernden Ausgaben des Blowfish-Algorithmus besetzt worden sind.

Dieser Initialisierungsprozeß ist relativ aufwendig, muß jedoch für jeden Schlüssel nur einmal durchgeführt werden, denn das P-Array und die S-Boxen können für die folgenden Blöcke im Speicher behalten werden.[203]

---

[200] Vgl. ebenda, S. 195.
[201] Vgl. ebenda, S. 197.
[202] Vgl. ebenda, S. 197 f.

Blowfish ist auf 32-Bit-Mikroprozessoren mit ausreichend Cache-Speicher für das P-Array und die S-Boxen um ein vielfaches schneller als DES. Durch die relativ aufwendige Schlüsselexpansion ist er allerdings nur für Anwendungen geeignet, bei denen sich die Schlüssel nicht allzu häufig ändern.[204]

Blowfish ist nicht patentiert und ist inclusive Quellcode in C frei erhältlich. Das macht ihn besonders attraktiv für Software-Entwickler und hat dazu geführt, daß der Algorithmus sehr beliebt ist und mittlerweile in über 130 Softwareprodukten implementiert wurde, darunter auch die Java Cryptography Extension, sowie diverse UNIX-Derivate und Internet-Tools.[205]

4.1.3.2 Sicherheit von Blowfish

Blowfish wurde entworfen, um den seinerzeit neuesten kryptoanalytischen Erkenntnissen zu widerstehen. Dazu wurden diverse Maßnahmen getroffen. Die großen, schlüsselabhängigen S-Boxen sollten ausreichenden Schutz vor differentieller und linearer Kryptoanalyse bieten, da deren Struktur dem Kryptoanalytiker nicht bekannt sein kann, ohne daß er den Schlüssel kennt. Durch die Verwendung von vier S-Boxen werden Symmetrien vermieden, die sonst auftreten könnten, wenn mehrere Eingabewerte der S-Box identisch sind. Die Verknüpfung mit den S-Boxen garantiert, daß innerhalb von drei Iterationen jedes Bit des Chiffretexts von jedem Klartextbit und jedem Schlüsselbit abhängig ist. Alle zwei weiteren Runden ist dies mit den respektiven Eingabeblöcken erneut der Fall. Die Schlüsselexpansion mit Hilfe des Blowfish-Algorithmus garantiert eine kontinuierliche Veränderung der erzeugten Teilschlüssel (Einträge des P-Arrays und der S-Boxen) und schützt so vor Angriffen auf feste Teilschlüssel von Algorithmen. Außerdem wird so sichergestellt, daß jedes Bit der Teilschlüssel von jedem Bit des Schlüssels K abhängig ist. Durch die zeitaufwendige Generierung des P-Arrays und der S-Boxen wird der Aufwand für einen Brute-force-Angriff stark erhöht, da sie für jeden Schlüssel neu durchgeführt werden muß.[206]

Blowfish wurde seit seiner Veröffentlichung vielfach kryptoanalytisch untersucht. Die wichtigsten Resultate werden im folgenden kurz vorgestellt.

---

[203] Vgl. ebenda, S. 197.
[204] Vgl. ebenda, S. 194 f.
[205] Vgl. **Counterpane Systems:** „Products That Use Blowfish“, Online im Internet, http://www.counterpane.com/products.html, Stand: unbekannt, Abruf: 11.01.2001.
[206] Vgl. **Schneier, Bruce**, 1994, a.a.O., S. 194 u. 198 ff.

JOHN KELSEY gelang es, einen reduzierten Blowfish mit nur drei Iterationen zu brechen, er konnte seinen Angriff jedoch nicht auf mehr Iterationen ausweiten.[207]

SERGE VAUDENAY hat eine reduzierte Blowfish-Variante untersucht, bei der die S-Boxen nicht schlüsselabhängig, sondern vorgegeben und bekannt waren. Er zeigte, daß es in diesem Falle für weniger als acht Iterationen möglich ist, das P-Array (der einzige Teil des Algorithmus, der hier noch schlüsselabhängig ist) durch differentielle Kryptoanalyse zu rekonstruieren. Ab acht Iterationen ist dieser Angriff unmöglich, da mehr als 2^{64} Klartexte nötig wären, was bei einer Blockgröße von 64 Bit unmöglich ist.[208]

Er fand auch heraus, daß bestimmte schwache Schlüssel schwache S-Boxen erzeugen, in denen zwei Einträge identisch sind. In diesem Falle wird der Angriff bei bekannten S-Boxen einfacher.[209] Bei schlüsselabhängigen S-Boxen und verringerter Iterationszahl kann festgestellt werden, daß ein schwacher Schlüssel verwendet wurde, nicht jedoch, wie dieser aussieht. Bei einem vollen Blowfish mit 16 Iterationen können die schwachen Schlüssel überhaupt nicht mehr ausgenutzt werden.[210]

Eine Stärke von Blowfish ist die variable Schlüssellänge von bis zu 448 Bit, die es ermöglicht, das Sicherheitsniveau an die individuellen Anforderungen anzupassen. Bei vollen 448 Bit Schlüssellänge dürfte ein Brute-force-Angriff für alle Zukunft unmöglich bleiben.

Obwohl für eine endgültige Beurteilung der Sicherheit von Blowfish noch weitere Analysen gefragt sind, scheint es dennoch angemessen, Blowfish als praktisch sicher einzustufen, wenn

- der Algorithmus mit der vollen Iterationszahl implementiert wird und
- der Schlüssel lang genug gewählt wird.

---

[207] Vgl. **Schneier, Bruce:** „The Blowfish Encryption Algorithm – One Year Later“, in: Dr. Dobb’s Journal, 20. Jg. (1995), Heft 9, Online im Internet, http://www.counterpane.com/bfdobsoyl.html, Stand: unbekannt, Abruf: 8.1.2001.

[208] Vgl. **Vaudenay, Serge:** „On the Weak Keys of Blowfish“, in: Fast Software Encryption – Third International Workshop Proceedings, **Hrsg.: Gollmann, Dieter**, Berlin et al. 1996, S. 30 f.

[209] Vgl. ebenda, S. 28 ff.

[210] Vgl. ebenda, S. 31.

4.2 Asymmetrische Algorithmen

4.2.1 RSA

4.2.1.1 Beschreibung des Algorithmus

Der Algorithmus RSA wurde 1978 von RONALD L. RIVEST, ADI SHAMIR und LEONARD ADLEMAN vom MIT Laboratory for Computer Science vorgestellt, nach denen er auch benannt ist. RSA war der erste vollwertige Public-key-Algorithmus und ist für die Chiffrierung und für digitale Signaturen gleichermaßen geeignet. Er erfreut sich seither großer Beliebtheit, was hauptsächlich darin begründet liegt, daß er von allen Public-key-Algorithmen am einfachsten zu implementieren ist. Außerdem hat man in den vielen Jahren der Nutzung von RSA sehr positive Erfahrungen damit gemacht.[211]

Zur Ermittlung der Schlüssel müssen zunächst zwei große Primzahlen, p und q, gewählt werden.[212] Diese sollten zwischen 100 und 200 Dezimalstellen lang sein. Für optimale Sicherheit empfiehlt es sich, darauf zu achten, daß die beiden Längen in etwa gleich sind.[213] Aus p und q errechnet sich durch Multiplikation n, der sogenannte *Modulus*:

$$n = p * q.$$[214]

Danach wird der Dechiffrierungsschlüssel d als große, ganze Zahl zufällig ausgewählt. d muß relativ prim zu dem Produkt $(p - 1)(q - 1)$ sein, das heißt der größte gemeinsame Teiler von d und $(p - 1)(q - 1)$ muß gleich 1 sein. Sodann kann der Chiffrierungsschlüssel e aus der Identität

$$e * d \equiv 1 \bmod ((p - 1)(q - 1))$$

durch Berechnung von

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

abgeleitet werden.[215] d^{-1} ist die multiplikative Inverse mod $((p - 1)(q - 1))$ zu d und kann mit Hilfe einer Variante des Euklid'schen Algorithmus zur Ermittlung des

---

211 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 466 f.
212 Vgl. **Rivest, Ronald L.; Shamir, Adi; Adleman, Leonard:** „A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", in: Communications of the ACM, 21. Jg. (1978), Heft 2, S. 122.
213 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 467.
214 Vgl. **Rivest, Ronald L.; Shamir, Adi; Adleman, Leonard**, a.a.O., S. 122.

größten gemeinsamen Teilers bestimmt werden.[216] Die Primzahlen p und q sollten nach der Ermittlung von n, d und e gelöscht werden.[217]

Zur Chiffrierung wird der Klartext M als sehr große Binärzahl interpretiert und es wird

$$C = E(M) = M^e \bmod n$$

berechnet. Zur Dechiffrierung wird mit dem Chiffretext C analog unter Verwendung der Formel

$$D(C) = C^d \bmod n$$

vorgegangen. Folglich stellt das Paar (e, n) den öffentlichen Schlüssel dar, mit dem jedermann eine Nachricht chiffrieren kann. Das Paar (d, n) ist der private Schlüssel. Wird d geheimgehalten, so kann nur der Inhaber mit (e, n) chiffrierte Nachrichten dechiffrieren.[218]

Ein Dritter kann d nicht ohne weiteres aus (e, n) ableiten, da zu dessen Berechnung p und q, die Primfaktoren von n, bekannt sein müssen. Die Faktorisierung extrem großer Zahlen ist jedoch äußerst schwierig und wird bei Zahlen zwischen 100 und 200 Dezimalstellen rechnerisch undurchführbar.[219] Auch die Gleichung

$$M = \sqrt[e]{C} \bmod n$$

ist ohne Kenntnis von p und q nicht berechenbar. Das folgt aus einem Satz der Zahlentheorie von LEONHARD EULER.[220]

Es läßt sich mathematisch nachweisen, daß sowohl die Ermittlung von $(p-1)(q-1)$ ohne die Faktorisierung von n, als auch das Auffinden eines d', welches die Faktorisierung von n erlaubt, mindestens ebenso schwierig sind, wie die direkte Faktorisierung von n.[221] Damit ist RSA theoretisch fundiert.

Es ist allerdings nicht möglich, mathematisch nachzuweisen, daß es notwendig ist, die Faktorisierung des Modulus zu kennen, um einen RSA-Chiffretext ohne Kenntnis von d zu dechiffrieren. Allerdings ist noch kein Verfahren bekannt, welches dieses

---

215 Vgl. ebenda, S. 123.
216 Vgl. ebenda, S. 124.
217 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 467.
218 Vgl. **Rivest, Ronald L.; Shamir, Adi; Adleman, Leonard**, a.a.O., S. 122.
219 Vgl. ebenda, S: 122 f. u. 125.
220 Vgl. **Münzenberger, M.**, a.a.O., S. 16.
221 Vgl. **Rivest, Ronald L.; Shamir, Adi; Adleman, Leonard**, a.a.O., S. 125 f.

leisten könnte. Ebenso ist, wie bereits unter Abschnitt 3.5.2.2 erwähnt, nicht Nachzuweisen, daß die Faktorisierung großer Zahlen ein schwer lösbares Problem ist.[222]

Es ist aufgrund der Bestimmungsgleichung für e unerheblich, ob d oder e zur Chiffrierung als Exponent verwendet wird. In jedem Falle wird die Dechiffrierung mit dem jeweils anderen Wert als Exponent funktionieren.[223] Dieser Umstand wird sich im Zusammenhang mit digitalen Signaturen noch als nützlich erweisen. Vgl. dazu Abschnitt 5.3.1.1.

Ist die zu chiffrierende Nachricht M länger als n, so wird sie in gleichlange Blöcke $M_i < n$ unterteilt und dann blockweise chiffriert:

$$C_i = M_i^e \bmod n,$$

bzw. zur Dechiffrierung

$$M_i = C_i^d \bmod n.$$ [224]

Implementationen von RSA sind etwa um den Faktor 1.000 langsamer, als Implementationen von DES.[225]

RSA wurde von seinen Erfindern in den USA patentiert. Das Patent lief am 20. September 2000 aus, seither ist RSA von jedermann frei nutzbar.[226] Das macht den Algorithmus noch interessanter für den potentiellen Anwender.

4.2.1.2 Sicherheit von RSA

Die Sicherheit von RSA beruht (aller Wahrscheinlichkeit nach) auf der Schwierigkeit, den Modulus n zu faktorisieren. Folglich wird RSA prinzipiell sicher bleiben, solange nicht eine neue, revolutionäre Methode zur Faktorisierung großer Zahlen entdeckt wird.[227] Der momentane Stand der Faktorisierungstechnologie liegt bei 155 Dezimalstellen mit erheblichem weltweitem Rechnereinsatz (Vgl. hierzu Abschnitt 3.5.2.2).[228] Je nach Sicherheitsbedürfnis sollte die Länge des Modulus also in etwa in diesem Bereich oder darüber liegen. Das deckt sich mit der Forderung nach Moduli von 1024 – 2048 Bit Länge aus Abschnitt 3.5.2.2.

---

222 Vgl. **Buchmann, Johannes**, a.a.O., S. 118 f.
223 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 467.
224 Vgl. **Münzenberger, M.**, a.a.O., S. 37 f.
225 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 469.
226 Vgl. ebenda, S, 474.
227 Vgl. ebenda, S. 470.
228 Vgl. **Leiberich, Otto**, a.a.O., S. 16 f.

Ein Brute-force-Angriff, in dem jeder mögliche Wert von d ausprobiert wird ist noch ineffizienter, als die Faktorisierung von n.[229] Es wurde auch nachgewiesen, daß der Versuch, einzelne Klartextbits aus einem RSA-Chiffretext zu extrahieren, ebenso schwierig ist, wie die Faktorisierung von n.[230]

Der Algorithmus selber kann folglich als sicher angesehen werden, sofern die Primzahlen p und q groß genug sind. Es können aber Unsicherheiten aus dem mit der konkreten Implementation verbundenen Protokoll erwachsen.[231] Im folgenden werden die problematischen Situationen kurz vorgestellt.

Es wurde ein Protokoll vorgeschlagen, in welchem eine zentrale Schlüsselvergabestelle allen Benutzern denselben Modulus n zuordnet und nur die Exponenten d_i und e_i für jeden Benutzer i getrennt generiert. Das würde die Arbeit der Schlüsselvergabestelle sehr erleichtern, da nur einmal zwei gute Primzahlen p und q gesucht werden müßten. Diese Vorgehensweise ermöglicht aber einige Angriffsmöglichkeiten. Wird jemals ein und dieselbe Nachricht an zwei Benutzer verschickt, deren Chiffrierungsexponenten e_1 und e_2 relativ prim zueinander sind, so kann diese Nachricht dechiffriert werden, ohne daß einer der Dechiffrierungsexponenten d_1 oder d_2 bekannt ist. Damit ist das Kryptosystem noch nicht gebrochen, aber bestimmte Nachrichten sind unsicher. Allerdings gibt es noch zwei weitere Angriffsmöglichkeiten in einer Umgebung mit gemeinsamem Modulus für alle Benutzer. Zum einen existiert ein probabilistischer Algorithmus, mit dem bei einem gegebenen Exponentenpaar d_1 und e_1 die Faktorisierung des Modulus n mit einem Erwartungswert von nur zwei Durchläufen gefunden werden kann. Somit kann jeder Benutzer den gemeinsamen Modulus faktorisieren. Ist die Faktorisierung des Modulus bekannt, kann wie unter Abschnitt 4.2.1.1 beschrieben zu jedem öffentlichen Chiffrierungsexponenten e_i der zugehörige Dechiffrierungsexponent d_i ermittelt werden. Der zweite Angriff beruht auf einem deterministischen Algorithmus, mit dem es möglich ist, bei einem bekannten Exponentenpaar d_1 und e_1 andere Exponentenpaare zu ermitteln, ohne daß die Faktorisierung des Modulus n bekannt sein muß. So kann auch jeder Benutzer die privaten Exponenten d_i der anderen Benutzer ermitteln. In den beiden letzteren Fällen ist das Kryptosystem tatsächlich gebrochen. Die Vergabe eines gemeinsamen Modulus an

[229] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 470.
[230] Vgl. **Alexi, Werner; Chor, Benny; Goldreich, Odad; Schnorr, Claus P.:** „RSA and Rabin Functions: Certain Parts are as Hard as the Whole“, in: SIAM Journal on Computing, 17. Jg. (1988), Heft 2, S. 203 f.
[231] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 473.

alle Benutzer eines Kommunikationssystems ist folglich nicht zu empfehlen, da diese Vorgehensweise Wege zum Bruch des Kryptosystems eröffnet.[232]

Viele Implementationen von RSA benutzen kleine Werte für den Chiffrierungsexponenten e des öffentlichen Schlüssels, da sich so die Chiffrierungsoperationen beschleunigen lassen. Jeder Benutzer des Systems wählt zwei große Primzahlen p_i und q_i und veröffentlicht deren Produkt n, sowie den Exponenten e_i. Wählen alle Benutzer kleine Werte für e_i, so ist die Wahrscheinlichkeit groß, daß mehrere Benutzer denselben Exponenten verwenden.[233] Dann ist es möglich, eine Nachricht nur mit Hilfe der Chiffretexte und der Moduli der betreffenden Benutzer zu dechiffrieren, wenn sie an mindestens e Benutzer geschickt wird, die den Exponenten e verwenden.[234] Sind die versendeten Nachrichten nur geringfügig unterschiedlich (z.B. bei Anfügung eines Zeitstempels oder unterschiedlicher Anrede in einem Serienbrief), so ist die Dechiffrierung noch immer möglich, wenn mehr als $e * (e + 1) / 2$ solcher Nachrichten an Benutzer mit demselben Exponenten e versandt werden.[235] Niedrige Chiffrierungsexponenten sind folglich problematisch und sollten nicht verwendet werden.[236] Will man die Geschwindigkeitsvorteile von niedrigen Chiffrierungsexponenten nicht verlieren, können auch alternativ alle Nachrichten vor der Chiffrierung mit unabhängigen Zufallsdaten aufgefüllt werden, bis sich auch identische Nachrichten signifikant unterscheiden. Die meisten praktischen Implementationen von RSA gehen so vor. [237]

Auch die Dechiffrierung läßt sich beschleunigen, wenn ein niedriger Exponent d verwendet wird.[238] Für diesen Fall existiert jedoch ein möglicher Angriff, der von MICHAEL J. WIENER entdeckt wurde. Er stellte einen Algorithmus vor, mit dem es möglich ist, mit Hilfe des öffentlichen Schlüssels (e, n) den privaten Dechiffrierungsexponenten d zu ermitteln, wenn dieser unter sonst normalen Bedingungen maximal etwa ein Viertel der Bitlänge des Modulus n aufweist und e kleiner als n ist.[239]

---

[232] Vgl. **Moore, Judy H.:** „Protocol Failures in Cryptosystems“, in: Proceedings of the IEEE, 76. Jg. (1988), Heft 5, S. 595 ff.
[233] Vgl. ebenda, S. 597.
[234] Vgl. **Hastad, Johan:** „On Using RSA with Low Exponent in a Public Key Network“, in: Advances in Cryptology – CRYPTO '85 Proceedings, **Hrsg.: Williams, Hugh C.**, Berlin et al. 1986, S. 403.
[235] Vgl. ebenda, S. 407.
[236] Vgl. **Moore, Judy H.**, a.a.O., S. 597.
[237] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 472 f.
[238] Vgl. **Wiener, Michael J.:** „Cryptanalysis of Short RSA Secret Exponents“, in: IEEE Transactions on Information Theory, 36. Jg. (1990), Heft 3, S. 553.
[239] Vgl. ebenda, S. 556.

Der Exponent d darf also nicht zu kurz gewählt werden. Dieser Fall tritt allerdings äußerst selten auf, wenn d und e zufällig gewählt werden und ist unmöglich, wenn e klein gewählt wird.[240]

Zusammenfassend kann gesagt werden, daß RSA-Kryptosysteme als praktisch sicher gelten können, wenn die folgenden Punkte beachtet werden:

- Die Moduli n_i dürfen nicht zu kurz gewählt werden. Vgl. Tab. 2 des Anhangs.
- Es sollte nicht mehreren Benutzern ein gemeinsamer Modulus zugeordnet werden.
- Die Chiffrierungsexponenten e_i sollten nicht zu klein gewählt werden oder versendete Nachrichten sollten mit Zufallsdaten aufgefüllt werden.
- Die Dechiffrierungsexponenten d_i sollten groß sein, keinesfalls kleiner als ein Viertel des Modulus n_i.

Wird RSA sowohl zur Chiffrierung von Nachrichten, als auch für digitale Signaturen verwendet, so gibt es einige weitere Punkte, die zu beachten sind. Diese werden im Abschnitt 5.3.1.2 erläutert, der sich mit der Sicherheit von RSA-Signaturen auseinandersetzt.

4.2.2 ElGamal

4.2.2.1 Beschreibung des Algorithmus

TAHER ELGAMAL schlug 1984 einen Public-key-Algorithmus vor, der auf dem Problem der diskreten Logarithmen in endlichen Körpern Modulo p beruht. Der Algorithmus ist sowohl zur Chiffrierung von Nachrichten, als auch für digitale Signaturen geeignet.[241]

Es sei p eine Primzahl. Ist g eine Primitivwurzel mod p, so existiert zu jedem $y \in \{1, 2, \dots, p-1\}$ ein Exponent $x \in \{1, 2, \dots, p-2\}$, für den gilt

$$y = g^x \bmod p.$$

---

[240] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 473.

[241] Vgl. **ElGamal, Taher:** „A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms“, in: IEEE Transactions on Information Theory, 31. Jg. (1985), Heft 4, S. 469 f.

Der Exponent x ist der *diskrete Logarithmus* von y zur Basis g. Die Ermittlung des diskreten Logarithmus gilt als schwieriges Problem. Dieser Umstand wird von ElGamal ausgenutzt.[242]

Zur Erzeugung eines Schlüsselpaares werden zunächst drei Zahlen ausgewählt: eine Primzahl p, eine Primitivwurzel mod p, genannt g, sowie eine Zufallszahl x, die kleiner als p sein muß. Sodann wird

$$y = g^x \bmod p$$

berechnet. Der öffentliche Schlüssel ist (y, g, p), der private Schlüssel ist x. Die Ableitung des privaten Schlüssels aus dem öffentlichen Schlüssel entspricht also der Berechnung eines diskreten Logarithmus. Dabei muß p so gewählt werden, daß $p - 1$ mindestens einen großen Primfaktor hat. Hat $p - 1$ nur kleine Primfaktoren, so ist der diskrete Logarithmus schnell zu ermitteln.[243]

Zur Chiffrierung der Nachricht M mit dem öffentlichen Schlüssel wird eine Zufallszahl k ausgewählt, die geheim gehalten werden muß. Dann werden

$$C_1 = g^k \bmod p \quad \text{und}$$
$$C_2 = y^k M \bmod p$$

berechnet. Das Paar (C_1, C_2) ist der Chiffretext. Der Chiffretext ist bei ElGamal folglich doppelt so groß, wie der Klartext. Damit dies funktioniert, muß $M < p - 1$ gelten. Ist M zu groß, so müssen Blöcke M_i gebildet werden, die dann einzeln chiffriert werden. Dabei sollte für jeden Block ein neuer Wert für k verwendet werden, da sonst ein Angreifer, dem es gelingt den Klartext zu einem Block zu ermitteln, alle anderen Blöcke leicht dechiffrieren kann.[244]

Zur Dechiffrierung von (C_1, C_2) mit dem privaten Schlüssel wird

$$M = \frac{C_2}{C_1^x \bmod p}$$

berechnet.[245]

Implementationen von ElGamal arbeiten etwa um den Faktor 10 langsamer als Implementationen von RSA.[246]

---

[242] Vgl. **Buchmann, Johannes**, a.a.O., S. 127 u. 130.
[243] Vgl. **ElGamal, Taher**, a.a.O., S. 469 f.
[244] Vgl. ebenda, S. 469.
[245] Vgl. ebenda, S. 469.

Obwohl ElGamal selber nicht patentiert wurde, war das US-Patentamt der Ansicht, daß ElGamal von einem Patent von WHITFIELD DIFFIE und MARTIN E. HELLMAN mit abgedeckt würde. Dieses Patent lief am 29. April 1997 aus. Damit war ElGamal der erste freie Public-key-Algorithmus, der sowohl zur Chiffrierung, als auch für digitale Signaturen geeignet ist.[247]

4.2.2.2 Sicherheit von ElGamal

Das ElGamal-Verfahren kann gebrochen werden, wenn es einem Angreifer gelingt, aus (C_1, C_2) und y entweder x oder k zu ermitteln. Da x und k nur in Exponenten vorkommen, entspricht dies der Berechnung von diskreten Logarithmen, was bei korrekter Auswahl der Parameter, wie oben beschrieben, schwierig ist.[248] Damit die Berechnung rechnerisch undurchführbar ist, sollte die Primzahl p mindestens 512 Bit, besser aber 768 – 1024 Bit lang sein. Bei Längen unterhalb 512 Bit können mit modernen Algorithmen zur Berechnung diskreter Logarithmen Ergebnisse erzielt werden. Außerdem muß k für jede Verschlüsselung neu bestimmt werden, da ein Angreifer, dem ein Klartext-Chiffretext-Paar zur Verfügung steht, leicht jeden anderen Chiffretext dechiffrieren kann, bei dem derselbe Wert für k verwendet wurde.[249]

Da der Wert k immer neu und zufällig gewählt wird, wird derselbe Klartext nicht zweimal denselben Chiffretext ergeben, was die Kryptoanalyse mit statistischen Verfahren erschwert.[250]

Das ElGamal-Verfahren läßt sich verallgemeinern, so daß es z.B. auch mit elliptischen Kurven arbeiten kann. Das ist ein Vorteil, falls einmal ein Algorithmus gefunden wird, der diskrete Logarithmen in endlichen Körpern Modulo p effizient berechnen kann.[251]

Das ElGamal-Verfahren kann folglich als praktisch sicher eingestuft werden, solange

- p groß genug gewählt wird und
- k für jede Chiffrierung zufällig neu gewählt wird.

---

246 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 470 u. 478.
247 Vgl. ebenda, S. 479.
248 Vgl. **ElGamal, Taher**, a.a.O., S. 469 f.
249 Vgl. **Buchmann, Johannes**, a.a.O., S. 132 f.
250 Vgl. ebenda, S. 133.
251 Vgl. ebenda, S. 133 f.

Das gilt, so lange kein effizienter Algorithmus zur Berechnung von diskreten Logarithmen existiert.

5 Verfahren zur Gewährleistung der Verbindlichkeit

5.1 Rechtliche Aspekte

Wann immer im Rechtsverkehr Willenserklärungen – seien es einseitige Erklärungen bzw. Versprechen, die Zustimmung zu Verträgen oder elektronische Banküberweisungen – auf elektronischem Wege ausgetauscht oder gespeichert werden, ist es von äußerster Wichtigkeit, daß diese rechtlich verbindlich und gegebenenfalls auch einklagbar sind.[252]

Im traditionellen Rechtsverkehr werden verbindliche Erklärungen zumeist durch handschriftliche Unterschriften auf Schriftstücken bekundet.[253] Der handschriftlichen Unterschrift wird in Verbindung mit der Schriftform eine Reihe von Eigenschaften allgemein zuerkannt.[254] Sie lauten wie folgt:[255]

1. **Authentizität:** Die Unterschrift überzeugt den Leser, daß der Unterzeichner sie willentlich geleistet hat.
2. **Integrität:** Ein schriftliches Dokument kann nachträglich nicht spurlos verändert werden.
3. **Fälschungssicherheit:** Nur der Unterzeichner kann seine eigene Unterschrift leisten, andere können sie nicht imitieren.
4. **Verifizierbarkeit:** Jeder Dritte ist in der Lage, die Echtheit einer Unterschrift zu überprüfen.
5. **Nachrichtenabhängigkeit:** Die Unterschrift ist Teil des unterzeichneten Dokuments und kann nicht auf ein anderes Dokument übertragen werden.
6. **Verbindlichkeit:** Der Unterzeichner kann im Nachhinein nicht abstreiten, die Unterschrift geleistet zu haben.

[252] Vgl. **Oppliger, Rolf**, a.a.O., S. 119.
[253] Vgl. **Bizer, Johann; Miedbrodt, Anja**, a.a.O. S. 136 f.
[254] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 34 f.
[255] Vgl. **Oppliger, Rolf**, a.a.O., S. 120 und **Schneier, Bruce**, 1996, a.a.O., S. 35, sowie **Bizer, Johann; Miedbrodt, Anja**, a.a.O., S. 137.

In der Realität gelten diese Eigenschaften nicht uneingeschränkt. Unterschriften können gefälscht oder von einem Dokument auf ein anderes übertragen werden. Auch können Dokumente nachträglich noch verändert werden. Man vertraut jedoch trotzdem der Unterschrift, da der Betrug hier sehr schwierig ist und von unbedarften Personen somit nicht vorgenommen werden kann, und weil es Möglichkeiten gibt, Manipulationen wie nachträgliche Änderungen oder gefälschte Unterschriften nachzuweisen.[256]

Elektronische Daten können jedoch von jedermann problemlos und ohne Spuren zu hinterlassen geändert oder kopiert werden. Der Empfänger eines elektronischen Dokuments kann also nicht sicher sein, daß das Dokument ihn unverfälscht erreicht hat und ob es tatsächlich vom angeblichen Urheber stammt.[257] Elektronische Dokumente können damit die Eigenschaften der Schriftform nicht ohne weiteres erfüllen.[258] Aus diesem Grunde sind Verfahren notwendig, die auch für elektronische Dokumente Authentizität, Integrität, etc. herstellen.[259] Solche Verfahren werden als digitale Signaturen bezeichnet und sollen im folgenden Abschnitt erläutert werden.

5.2 Digitale Signaturen

Die im vorigen Abschnitt genannten sechs Eigenschaften der handschriftlichen Unterschriften auf physikalischen Dokumenten können als die Kriterien angesehen werden, die digitale Signaturen zu erfüllen haben, damit sie rechtliche Verbindlichkeit herstellen können.

Das einfache Anhängen einer irgendwie gearteten Signatur an ein elektronisches Dokument reicht dazu nicht aus, da dieses Dokument trotzdem von jedermann geändert werden könnte. Außerdem könnte die Signatur problemlos an ein anderes Dokument angehängt werden.[260] Folglich müssen digitale Signaturen komplexere Gebilde sein, um den genannten Anforderungen entsprechen zu können. Neben der Funktion einer reinen Unterschrift müssen sie auch die Funktion eines Siegels übernehmen.[261]

---

[256] Vgl. **Buchmann, Johannes**, a.a.O., S. 175, sowie **Schneier, Bruce**, 1996, a.a.O., S. 35.
[257] Vgl. **Bizer, Johann; Miedbrodt, Anja**, a.a.O., S. 141.
[258] Vgl. ebenda, S. 139.
[259] Vgl. ebenda, S. 141.
[260] Vgl. **Buchmann, Johannes**, a.a.O., S. 175, sowie **Schneier, Bruce**, a.a.O., 1996, S. 35.
[261] Vgl. **Heuser, Ansgar**, a.a.O., S. 12.

Kryptografische Verfahren in Verbindung mit einem geeigneten Protokoll bieten sich zur Lösung dieser Problematik an. Der Besitz eines geheimen Schlüssels kann so als Substitut für eine handschriftliche Unterschrift dienen. Die Idee ist, daß jemand eine Nachricht chiffriert und aus der Verschlüsselung ersichtlich wird, daß nur er die Nachricht chiffriert haben kann. Die Benutzung von symmetrischen Algorithmen für diese Aufgabe würde zu einer extrem komplizierten Schlüsselinfrastruktur führen, da jeder Schlüssel nur einem Sender und einem Empfänger bekannt sein darf.[262] Deshalb werden digitale Signaturen grundsätzlich mit Public-key-Algorithmen erzeugt. Diese werden dabei in einer Weise benutzt, daß der Dechiffrierungsschlüssel veröffentlicht wird und der Chiffrierungsschlüssel nur dem Unterzeichner bekannt ist.[263] Dies ist genau die umgekehrte Vorgehensweise, wie in Abschnitt 3.5.2.1 beschrieben wurde, wo der Chiffrierungsschlüssel öffentlich und der Dechiffrierungsschlüssel geheim ist.

Es können für digitale Signaturen speziell für diesen Zweck entworfene Public-key-Algorithmen verwendet werden. Einige Public-key-Algorithmen, die in der unter Abschnitt 3.5.2.1 beschriebenen Weise zur Verschlüsselung von Nachrichten eingesetzt werden, können jedoch auch für digitale Signaturen genutzt werden. Das sind ElGamal, Rabin und RSA.[264]

Es gibt prinzipiell zwei Möglichkeiten, eine Nachricht digital zu signieren:[265]

1. Die Nachricht wird vom Unterzeichner in voller Länge mit seinem privaten Schlüssel chiffriert.
2. Der Unterzeichner bildet einen Hash-Wert der zu signierenden Nachricht mit Hilfe einer allgemein bekannten, kollisionsresistenten Hash-Funktion, chiffriert diesen mit seinem privaten Schlüssel und hängt ihn an den Klartext der Nachricht an.

Im ersten Fall kann jedermann durch Dechiffrierung mit dem öffentlichen Schlüssel des Unterzeichners die Signatur verifizieren. Die Nachricht kann auch nicht manipuliert werden, da niemand außer dem Unterzeichner dessen privaten Schlüssel kennt.[266] Im zweiten Fall wird zur Verifizierung der angehängte Hash-Wert mit dem

[262] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 35 f.
[263] Vgl. ebenda, S. 39.
[264] Vgl. ebenda, S. 461.
[265] Vgl. **Oppliger, Rolf**, a.a.O., S. 120 f.
[266] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 37 f.

öffentlichen Schlüssel des Unterzeichners dechiffriert. Dann wird mit derselben Hash-Funktion ein neuer Hash-Wert über das Dokument gebildet und mit dem dechiffrierten Wert verglichen. Sind die beiden Werte verschieden, ist entweder die Signatur falsch oder das Dokument wurde verändert.[267]

Qualitativ sind beide Protokolle folglich gleichwertig. Das Letztere bietet jedoch einige Vorteile. Die Signatur kann vom Dokument getrennt gehalten werden. Müssen mehrere Personen eine Nachricht signieren, so ist dies hier einfacher, da die Signaturen unabhängig sind und einzeln verifiziert werden können. Bei langen Nachrichten, wie z.B. größeren Vertragswerken, sind Public-key-Algorithmen zu langsam, um sie komplett zu verschlüsseln. Die Signierung mit Hash-Werten löst auch dieses Problem.[268]

Es müssen allerdings noch einige weitere Probleme beseitigt werden, um digitale Signaturen sicher zu machen. So könnte z.B. jemand eine von jemand anders signierte Nachricht kopieren und wiederholt versenden. Dies ist oftmals unschädlich, kann aber auch fatal sein, z.B. bei digitalen Schecks. Hier könnte jemand durch wiederholtes Einlösen von Kopien desselben Schecks beliebig hohen geldlichen Schaden anrichten. Dies kann dadurch vermieden werden, daß der Signatur ein *Zeitstempel*, also die genaue Tageszeit und das Datum, an dem die Signatur geleistet wurde, hinzugefügt wird. Wenn im Beispiel die Bank die Zeitstempel der eingelösten Schecks speichert und mehrere identische Schecks mit gleichen Zeitstempeln nicht akzeptiert, ist das Problem gelöst.[269]

Darüber hinaus muß es für denjenigen, der eine Signatur verifizieren möchte, eine Möglichkeit geben, sicherzugehen, daß er den korrekten öffentlichen Schlüssel dafür besitzt. Ansonsten könnte beispielsweise ein bösgläubiger Dritter sich als jemand anders ausgeben, einen falschen Schlüssel in Umlauf bringen und dann im Namen dieser Person Dokumente signieren. Oder ein Unterzeichner könnte später ablehnen, die Signatur geleistet zu haben. Die Lösung für dieses Problem ist eine unabhängige, vertrauenswürdige *Zertifizierungsstelle*.[270]

---

[267] Vgl. ebenda, S. 38 f.
[268] Vgl. ebenda, S. 38 ff.
[269] Vgl. ebenda, S. 38.
[270] Vgl. **Grant, Gail L.:** Understanding Digital Signatures – Establishing Trust over the Internet and other Networks, New York et al. 1997, S. 36.

Die Kommunikationsteilnehmer müssen sich bei der Zertifizierungsstelle anmelden und ein *Zertifikat* beantragen. die Zertifizierungsstelle überprüft daraufhin die Identität des Antragstellers, stellt gegebenenfalls ein Zertifikat aus und generiert ein Schlüsselpaar für den Antragsteller. Das Zertifikat enthält Namen und Organisation der betreffenden Person, deren öffentlichen Schlüssel, Informationen über die vom Unterzeichner verwendeten Algorithmen und das Ablaufdatum des Zertifikats. Das Zertifikat wird von der Zertifizierungsstelle signiert, um sicherzustellen, daß es authentisch und unverändert ist. Jeder Benutzer kann Zertifikate verifizieren, solange sichergestellt ist, daß jeder Benutzer den öffentlichen Schlüssel der Zertifizierungsstelle kennt. Durch das Zertifikat ist nun der öffentliche Schlüssel jedes Unterzeichners mit seiner Identität verbunden.[271] Die Zertifizierungsstellen sind darüber hinaus in der Lage, Zertifikate zu sperren (z.B. bei Verlust des privaten Schlüssels einer Person) und Statusinformationen über Zertifikate zu erteilen. Dazu werden ständig aktualisierte Sperrlisten gehalten.[272] Außerdem versehen sie auf Verlangen der Benutzer Dokumente mit Zeitstempeln, die allgemein akzeptiert werden können.[273]

Optimalerweise sollten die von den Zertifizierungsstellen generierten privaten Schlüssel auf sicheren Datenträgern, wie z.B. Chipkarten, an die Benutzer übergeben werden. Die Übergabe sollte vorzugsweise persönlich und schriftlich bestätigt durchgeführt werden.[274]

Die Zertifizierungsstellen müssen natürlich ihrerseits durch höhere Instanzen legitimiert werden. Dies läuft letztlich auf eine staatliche Stelle als ultimativ vertrauenswürdigen Dritten hinaus. In der Bundesrepublik Deutschland ist dies beispielsweise die Regulierungsbehörde für Telekommunikation und Post.[275]

Will ein Sender eine digital signierte Nachricht an einen Empfänger übermitteln, verläuft das vollständige Protokoll also folgendermaßen:[276]

1. Vor der Kommunikation muß sich der Sender ordnungsgemäß bei der Zertifizierungsstelle anmelden um ein Zertifikat nebst Schlüssel ausgestellt zu bekommen.

---

[271] Vgl. ebenda, S. 36 f.
[272] Vgl. **Bizer, Johann; Miedbrodt, Anja**, a.a.O., S. 144.
[273] Vgl. ebenda, S. 149.
[274] Vgl. ebenda, S. 143 f. u. 148.
[275] Vgl. ebenda, S. 143 u. 147.
[276] Vgl. **Grant, Gail L.**, a.a.O., S. 38 f.

2. Der Sender erstellt ein Hash der Nachricht, fügt eventuell einen Zeitstempel bei und chiffriert diese Daten mit Hilfe eines Public-key-Algorithmus und seines privaten Schlüssels.
3. Der Sender hängt die so entstandene Signatur an die Nachricht an und sendet sie an den Empfänger.
4. Der Empfänger überprüft oder ermittelt anhand des Zertifikates den korrekten öffentlichen Schlüssel.
5. Der Empfänger überprüft, ob das Zertifikat noch gültig ist.
6. Der Empfänger dechiffriert die digitale Signatur mit dem erhaltenen öffentlichen Schlüssel.
7. Der Empfänger erstellt ein neues Hash über die empfangene Nachricht und vergleicht es mit dem aus der digitalen Signatur gewonnenen Hash.
8. Sind beide Hash-Werte identisch, so ist die Signatur gültig und das Dokument ist unverändert.

Eine solche digitale Signatur erfüllt die unter Abschnitt 5.1 genannten sechs Kriterien. Sie gewährleistet keine Vertraulichkeit, das kann aber durch Chiffrierung der Nachricht zusammen mit der Signatur mit Hilfe eines geeigneten kryptografischen Algorithmus geschehen.[277]

Die einzige Situation, der kein denkbares Protokoll vorbeugen kann, ist die, daß ein Unterzeichner vorgibt, sein privater Schlüssel sei kompromittiert worden und er damit geleistete Signaturen abstreitet.[278] Es kann nur versucht werden, die Auswirkungen eines solchen Verhaltens einzugrenzen, indem die Zertifizierungsstelle unverzügliche Meldung im Kompromittierungsfall verlangt[279] und Signaturen generell mit Zeitstempel angefertigt werden. In diesem Fall bleiben zumindest alle Signaturen, die vor der Meldung des Schlüsselverlustes geleistet wurden, gültig.[280]

Zur Schaffung eines administrativen Rahmens bezüglich der Beweissicherheit digitaler Signaturen hat der Bundesdeutsche Gesetzgeber 1997 im Rahmen des Gesetzes zur Regelung der Rahmenbedingungen für Informations- und Kommunikationsdienste (Informations- und Kommunikationsdienste-Gesetz) das Gesetz zur digitalen Sig-

[277] Vgl. ebenda, S. 35 f.
[278] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 40.
[279] Vgl. **Bizer, Johann; Miedbrodt, Anja**, a.a.O., S. 149.
[280] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 40.

natur (Signaturgesetz) mit der dazugehörigen Verordnung zur digitalen Signatur (Signaturverordnung) verabschiedet. Nach diesen Vorschriften angefertigte digitale Signaturen gelten rechtlich als fälschungssicher, die so signierten Daten als sicher vor Verfälschungen. Im Signaturgesetz wird die oben erläuterte Vorgehensweise zur Signierung festgeschrieben. Es werden weiterhin die Voraussetzungen für die Zulassung von Zertifizierungsstellen und deren Pflichten, sowie die Pflichten der Benutzer festgelegt. Als Lizensierungsstelle für Zertifizierungsstellen wird die Regulierungsbehörde für Telekommunikation und Post benannt. Außerdem werden Richtlinien für die verwendeten technischen Komponenten aufgestellt.[281]

Die Regulierungsbehörde für Telekommunikation und Post empfiehlt derzeit die Verwendung der Hash-Funktionen RIPEMD-160 oder SHA-1, jeweils mit einer Hash-Wert-Länge von 160 Bit. Zur Chiffrierung des Hash-Wertes werden die Public-key-Algorithmen RSA und DSA, sowie DSA-Varianten, die auf elliptischen Kurven basieren, empfohlen.[282]

Ein wichtiges Problem im Bereich des rechtlichen Rahmens für digitale Signaturen ist die fehlende internationale Übereinkunft. In Deutschland gilt das Signaturgesetz in Verbindung mit der Signaturverordnung, eine EU-Richtlinie ist in Vorbereitung, jedoch noch nicht verabschiedet[283] und in den USA gilt der Digital Signature Standard (DSS) des NIST im Zusammenhang mit entsprechender Legislation.[284] Das erschwert die Anerkennung digitaler Signaturen über Ländergrenzen hinweg und kann so internationale elektronische Geschäftsbeziehungen nicht unerheblich behindern.

Im folgenden werden die wichtigsten Algorithmen zur Erzeugung digitaler Signaturen vorgestellt. Für die Auswahl einer geeigneten Hash-Funktion wird auf die Abschnitte 6.2 und 7.4 verwiesen, die sich detailliert mit diesem Thema auseinandersetzen.

---

281 Vgl. **Bizer, Johann; Miedbrodt, Anja**, a.a.O., S. 147 ff.

282 Vgl. **Regulierungsbehörde für Telekommunikation und Post:** „Geeignete Kryptoalgorithmen gemäß § 17 (2) SigV“, Online im Internet, http://www.regtp.de/imperia/md/content/tech_reg_t/digisign/23.pdf, Stand 28.12.2000, Abruf 28.12.2000, S. 2 f.

283 Vgl. **Bizer, Johann; Miedbrodt, Anja**, a.a.O., S. 136.

284 Vgl. **National Institute of Standards and Technology:** FIPS-PUB 186-2, Digital Signature Standard (DSS), Online im Internet, http://csrc.nist.gov/publications/fips/fip186-2/fips186-2.pdf, Stand: 27.01.2000, Abruf: 05.01.2001, S. 1.

5.3 Algorithmen zur Erzeugung digitaler Signaturen

5.3.1 RSA

5.3.1.1 Funktionsweise von RSA-Signaturen

Wie bereits in Abschnitt 4.2.1.1 erwähnt, weist RSA eine äußerst praktische Reziprozität auf: die Chiffrierungs- und Dechiffrierungsfunktionen arbeiten jeweils mit dem öffentlichen und dem privaten Schlüssel gleichermaßen. Es kann also immer mit dem einen Schlüssel dechiffriert werden, was mit dem anderen chiffriert wurde. Wurde eine Nachricht mit dem öffentlichen Schlüssel chiffriert, kann sie mit dem dazugehörigen privaten Schlüssel dechiffriert werden. Wurde sie mit dem privaten Schlüssel chiffriert, so kann sie mit dem öffentlichen Schlüssel dechiffriert werden. Die Chiffrierungs- und Dechiffrierungsfunktionen müssen dabei nicht verändert werden. Das macht RSA für die Erzeugung von digitalen Signaturen besonders attraktiv.[285]

Ist, wie in Abschnitt 4.2.1.1 erläutert, der öffentliche Schlüssel eines Benutzers (e, n) und der private Schlüssel (d, n), so ist das Vorgehen bei der digitalen Signatur identisch zur Chiffrierung von Nachrichten, mit Ausnahme des Vertauschens der Schlüssel. Zur Signierung einer Nachricht M mit dem privaten Schlüssel (M kann dabei sowohl die Nachricht selber, als auch der Hash-Wert einer Nachricht sein) wird

$$S = M^d \bmod n$$

berechnet. Die Verifizierung der Unterschrift mit dem öffentlichen Schlüssel erfolgt durch Berechnung von

$$D(S) = S^e \bmod n.$$[286]

RSA ist in den USA vom NIST im Rahmen des Digital Signature Standards (DSS) als möglicher Signaturalgorithmus offiziell anerkannt worden.[287]

---

[285] Vgl. **Grant, Gail L**, a.a.O., S. 33.
[286] Vgl. **Denning, Dorothy E.:** „Digital Signatures with RSA and Other Public-Key Cryptosystems“, in: Communications of the ACM, 27. Jg. (1984), Heft 4, S. 388.
[287] Vgl. **National Institute of Standards and Technology**, 2000, a.a.O., S. 7.

5.3.1.2 Sicherheit von RSA-Signaturen

Da RSA-Signaturen sich methodisch nicht von RSA-Chiffrierungen unterscheiden, gilt für die Sicherheit von RSA-Signaturen generell das unter Abschnitt 4.2.1.2 zur Sicherheit von RSA ausgeführte analog.

Allerdings existieren Angriffsmöglichkeiten über Chosen-ciphertext-Angriffe, wenn in einem System für die Signierung die gesamte zu signierende Nachricht verschlüsselt wird (Signieren ohne Hash-Funktion).[288]

Wird in einem System RSA sowohl zur Chiffrierung von Nachrichten, als auch zur Erzeugung von digitalen Signaturen verwendet ist es einem Angreifer möglich, den Klartext M zu einem abgefangenen Chiffretext C zu ermitteln. Er wählt dazu zufällig eine Nachricht R aus und berechnet X, indem er R mit dem öffentlichen Schlüssel des betreffenden Empfängers chiffriert:

$$X = R^e \bmod n.$$

Dann berechnet er $Y = XC \bmod n$ und läßt sich Y vom Empfänger signieren:

$$S = Y^d \bmod n.$$

Mit Hilfe von $Z = R^{-1} \bmod n$ kann er nun durch Berechnung von $ZS \bmod n$ den Klartext M ermitteln, denn es gilt:

$$ZS \bmod n = R^{-1}Y^d \bmod n = R^{-1}X^dC^d \bmod n = C^d \bmod n = M.$$[289]

Dieser Angriff funktioniert offensichtlich nur, wenn zur Chiffrierung und zur Signierung dieselben Schlüsselpaare verwendet werden.[290]

Es ist einem Angreifer möglich, digitale Signaturen zu fälschen. Möchte er von einem Benutzer die Nachricht M unterzeichnet haben, so wählt er eine zufällige Nachricht R aus. Diese verschlüsselt er mit dem öffentlichen Schlüssel (e, n) des Opfers, der eigentlich zur Verifikation von Signaturen gedacht ist:

$$X = R^e \bmod n.$$

Danach läßt er das Opfer XM signieren:

$$S = (XM)^d \bmod n.$$

---

[288] Vgl. ebenda, S. 388.
[289] Vgl. ebenda, S. 389.
[290] Vgl. ebenda, S. 389.

Durch die Operation SR^{-1} mod $n = M^d$ mod n erhält der Angreifer die gewünschte Unterschrift, ohne daß das Opfer M je gesehen hat.[291]

Dieser Angriff funktioniert nicht, wenn statt der Nachricht selber ein Hash-Wert der Nachricht signiert wird.[292]

Werden Nachrichten in einem Protokoll chiffriert, bevor sie signiert werden, so ergibt sich eine weitere Angriffsmöglichkeit. Sendet Benutzer 1 an Benutzer 2 eine signierte Nachricht M, so wird er sie erst mit dem öffentlichen Schlüssel (e_2, n_2) von Benutzer 2 chiffrieren und dann mit seinem eigenen privaten Schlüssel (d_1, n_1) signieren. Das ergibt

$$\left(M^{e_2} \bmod n_2\right)^{d_1} \bmod n_1 .$$

Will nun Benutzer 2 behaupten, Benutzer 1 habe ihm die Nachricht M' signiert zugesandt, so ist dies möglich, indem er ein x findet, für das gilt:

$$M'^x = M \bmod n_2.$$

Das ist möglich, da er die Primfaktoren p und q seines eigenen Modulus n_2 kennt. Nun veröffentlicht er als neuen öffentlichen Schlüssel (xe_2, n_2) und kann damit behaupten, Benutzer 1 habe ihm M' anstatt von M zugesandt.[293] Dieser Angriff kann durch die Verwendung von Hash-Funktionen nicht verhindert werden, sondern nur, indem den Benutzern nicht erlaubt wird, ihre öffentlichen Schlüssel jederzeit nach Belieben zu ändern.[294]

Um sicherzugehen, daß keine Angriffe der oben beschriebenen Art stattfinden, sollten also zusammenfassend folgende Sicherheitsmaßnahmen zusätzlich zu den in Abschnitt 4.2.1.2 erläuterten Maßnahmen getroffen werden:

- Wird RSA zur Chiffrierung und zur Signierung verwendet, so sollten unterschiedliche Schlüsselpaare zum Einsatz kommen. Dies ist ohnehin sinnvoll, da so bei der Kompromittierung eines der beiden Schlüsselpaare das andere nicht in Mitleidenschaft gezogen wird[295].

---

[291] Vgl. **Moore, Judy H.**, a.a.O., S. 595.
[292] Vgl. **Denning, Dorothy E.**, a.a.O., S. 390 f.
[293] Vgl. **Anderson, Ross J.; Needham, Roger:** „Robustness Principles for Public Key Protocols“, in: Advances in Cryptology – CRYPTO '95 Proceedings, **Hrsg.: Coppersmith, Don**, Berlin et al. 1995, S. 237.
[294] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 473 f.
[295] Vgl. ebenda, S. 42.

- Die Signierung sollte mit Hilfe von Hash-Funktionen erfolgen.
- Eine Chiffrierung von Nachrichten sollte möglichst nach der Signierung erfolgen. Falls sie vorher erfolgt, dürfen Benutzer ihre öffentlichen Schlüssel nicht nach Belieben ändern.

Werden diese Einschränkungen beachtet, so sind Signaturen mit RSA praktisch sicher, ebenso wie mit RSA chiffrierte Nachrichten, wie unter Abschnitt 4.2.1.2 erläutert.

5.3.2 ElGamal

5.3.2.1 Funktionsweise von ElGamal-Signaturen

Beim ElGamal-Verfahren unterscheidet sich der Signierungsvorgang von der Chiffrierung von Nachrichten. Nur die Erzeugung der Schlüssel (y, g, p) (öffentlich) und x (privat) ist identisch mit der unter Abschnitt 4.2.2.1 angegebenen Vorgehensweise.[296]

Zur Signierung einer Nachricht M, die kleiner als $p - 1$ sein muß, wird zunächst ein Wert k zufällig ausgewählt. k muß relativ prim zu $p - 1$ sein. Dann wird

$$r = g^k \bmod p$$

berechnet und die Gleichung

$$M = (xr + ks) \bmod (p - 1)$$

mit Hilfe des Euklid'schen Algorithmus nach s aufgelöst. Es existiert immer eine Lösung für s, wenn k relativ prim zu $p - 1$ ist. Die Signatur ist das Paar (r, s).[297]

Um eine Signatur zu verifizieren, muß nur geprüft werden, ob die Gleichung

$$g^M = y^r r^s \bmod p$$

erfüllt ist.[298]

Das ElGamal Signaturverfahren läßt sich wie das Chiffrierverfahren verallgemeinern, so daß elliptische Kurven verwendet werden können.[299]

[296] Vgl. **ElGamal, Taher**, a.a.O., S. 470.
[297] Vgl. ebenda, S. 470.
[298] Vgl. ebenda, S. 470.
[299] Vgl. **Buchmann, Johannes**, a.a.O., S. 185.

5.3.2.2 Sicherheit von ElGamal-Signaturen

Wie die ElGamal-Chiffrierung beziehen auch ElGamal-Signaturen ihre Sicherheit aus der Schwierigkeit, diskrete Logarithmen in endlichen Körpern zu berechnen. Jedoch läßt sich hier nicht nachweisen, daß die Schwierigkeit, das Signaturschema zu brechen, der Berechnung von diskreten Logarithmen entspricht.[300]

Ein Angreifer könnte versuchen, x durch die Analyse mehrerer Nachricht-Signatur-Paare zu ermitteln. Durch die zufällige Ermittlung von k bei jeder Signatur führt dies jedoch zu einem undeterminierten Gleichungssystem, dessen Lösungsmenge sehr groß ist. Wird jedoch nur zweimal der gleiche Wert für k verwendet, so ist das Gleichungssystem eindeutig determiniert und x kann errechnet werden. Der Versuch, die Gleichung zur Ermittlung von s nach x aufzulösen, erfordert die Berechnung von diskreten Logarithmen und ist somit bei der unter Abschnitt 4.2.2.2 beschriebenen Parameterauswahl rechnerisch undurchführbar. Auch der Versuch, x durch die Entwicklung linearer Abhängigkeiten zwischen den Unbekannten zu ermitteln, ist äquivalent zur Berechnung diskreter Logarithmen, da die Lösung dieser Vorgehensweise die Ermittlung des diskreten Logarithmus ermöglichen würde.[301]

Es scheint keine Möglichkeit zu geben, die obigen Gleichungen zwecks Fälschung einer Signatur bei gegebener Nachricht M in sinnvollem Zeitrahmen nach r und s aufzulösen.[302] Es ist allerdings möglich, bei einem gegebenen Nachricht-Signatur-Paar weitere gültige Nachricht-Signatur-Paare zu erzeugen. Die Nachricht kann dabei aber nicht gewählt werden, sondern ergibt sich aus den Berechnungen. Sie wird daher aller Wahrscheinlichkeit nach keinen Sinn ergeben. Um dieses Vorgehen unmöglich zu machen, sollten nicht die Nachrichten selber, sondern nur Hash-Werte der Nachrichten signiert werden. Das Verfahren selber funktioniert zwar weiter, aber bei Verwendung einer kollisionsresistenten kryptografischen Hash-Funktion ist es für den Fälscher rechnerisch undurchführbar, eine Nachricht zu erzeugen, deren Hash-Wert mit dem Ergebnis seiner Berechnungen übereinstimmt.[303]

Es läßt sich auch nachweisen, daß es gelingt, bei Vorhandensein eines Nachricht-Signatur-Paares mit Hilfe komplizierter mathematischer Operationen weitere Signaturen für frei gewählte Nachrichten zu erzeugen, wenn der vorhandene Wert für r

---

300 Vgl. **ElGamal, Taher**, a.a.O., S. 470.
301 Vgl. ebenda, S. 470.
302 Vgl. ebenda, S. 470.
303 Vgl. **Buchmann, Johannes**, a.a.O., S. 183 f., sowie **ElGamal, Taher**, a.a.O., S. 470.

außerhalb des Intervalls $[1, p - 1]$ liegt. Folglich muß unbedingt bei der Wahl von k darauf geachtet werden, daß die Bedingung $1 \leq r \leq p - 1$ nicht verletzt wird.[304]

Der unter Abschnitt 5.3.1.2 beschriebene Angriff auf RSA-Signaturen, die nach der Chiffrierung geleistet werden, funktioniert in abgewandelter Form auch für ElGamal-Signaturen. Wählt ein Benutzer sein y so aus, daß es ihm möglich ist, diskrete Logarithmen zu berechnen, so kann er Signaturen fälschen. Die Lösung des Problems ist, jedem Benutzer den gleichen Wert für y zuzuteilen. Das kann aber problematisch sein. Deshalb sollte hier generell erst nach der Signatur chiffriert werden.[305]

Folglich können ElGamal-Signaturen als praktisch sicher angesehen werden, sofern

- p groß genug gewählt wird,
- k für jede Signatur neu bestimmt wird,
- die Bedingung $1 \leq r \leq p - 1$ gilt,
- bei Benutzung von ElGamal für Signaturen und Chiffrierung grundsätzlich erst nach der Signierung chiffriert wird, und
- Signaturen mit Hilfe von Hash-Funktionen erzeugt werden.

Das gilt, wie bei der Chiffrierung, so lange kein effizienter Algorithmus zur Berechnung von diskreten Logarithmen gefunden wird.

5.3.3 Digital Signature Algorithm (DSA)

5.3.3.1 Funktionsweise von DSA-Signaturen

Der Digital Signature Algorithm (DSA) wurde 1991 vom NIST im Rahmen des Digital Signature Standards (DSS) als unpatentierte Alternative zu RSA veröffentlicht.[306] Allerdings ist momentan noch unklar, ob DSA ein weltweites Patent von C. P. SCHNORR verletzt. Dieses Patent läuft erst im Jahre 2008 aus, was bei der Verwendung von DSA ein erhebliches Problem darstellen kann.[307]

---

304 Vgl. **Buchmann, Johannes**, a.a.O., S. 183 f.
305 Vgl. **Anderson, Ross J.; Needham, Roger**, a.a.O., S. 237 f.
306 Vgl. **National Institute of Standards and Technology**, 2000, a.a.O., S. 3 u. 7.
307 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 493 f.

DSA ist eine Variante des ElGamal-Signaturalgorithmus und bezieht seine Sicherheit folglich ebenfalls aus der Schwierigkeit der Berechnung diskreter Logarithmen.[308] Der Algorithmus ist für die Benutzung in Verbindung mit einer 160-Bit-Hash-Funktion konzipiert, im Digital Signature Standard ist dafür SHA-1 vorgesehen.[309] Die Hash-Funktion wird hier mit *H(M)* bezeichnet.

Zur Schlüsselgenerierung müssen die folgenden Parameter bestimmt werden:[310]

- Eine Primzahl p, für die gilt $2^{L-1} < p < 2^L$, mit $512 < L < 1024$ und L ein Vielfaches von 64 (p ist also zwischen 512 und 1024 Bit lang),
- q, ein Primfaktor von $p - 1$ von der Länge 160 Bit,
- $g = h^{(p-1)/q} \bmod p$, mit h als beliebige ganze Zahl, für die gilt $g > 1$,
- eine Zufallszahl x die kleiner ist, als q,
- $y = g^x \bmod p$.

Der private Schlüssel ist x, der öffentliche Schlüssel ist (p, q, g, y).[311]

Um eine Nachricht M zu signieren wird eine Zufallszahl k ausgewählt, die kleiner als q sein muß. Sodann wird

$$r = (g^k \bmod p) \bmod q \quad \text{und}$$
$$s = (k^{-1}\,(H(M) + xr)) \bmod q$$

berechnet. Dabei ist k^{-1} die multiplikative Inverse mod q von k. Das Paar (r, s) ist die Signatur. Der Wert k muß für jede Signatur neu gewählt werden.[312]

Die Verifikation der Signatur erfolgt durch Berechnung von

$$w = s^{-1} \bmod q,$$
$$u_1 = (H(M) * w) \bmod q,$$
$$u_2 = (rw) \bmod q \quad \text{und}$$
$$v = ((g^{u_1} * y^{u_2}) \bmod p) \bmod q.$$

Die Signatur ist gültig, wenn $v = r$ gilt.[313]

---

[308] Vgl. ebenda, S. 486.
[309] Vgl. **National Institute of Standards and Technology**, 2000, a.a.O., S. 7 f.
[310] Vgl. ebenda, S. 8.
[311] Vgl. ebenda, S. 8.
[312] Vgl. ebenda, S. 8 f.
[313] Vgl. ebenda, S. 9.

Auch DSA läßt sich verallgemeinern, um elliptische Kurven verwenden zu können. Diese Variante wird vom NIST als ECDSA bezeichnet.[314]

Die Schlüsselgenerierung von DSA läuft schneller, als bei RSA, die Signierung ist etwa gleich schnell. Die Verifikation hingegen ist um einen Faktor zwischen 10 und 40 aufwendiger, abhängig von den Parameterwerten.[315]

5.3.3.2 Sicherheit von DSA-Signaturen

Durch den Zwang zur Benutzung einer Hash-Funktion bei der Signierung werden einige Sicherheitsprobleme, die bei RSA und ElGamal ohne Verwendung einer Hash-Funktion auftreten, von vornherein vermieden.[316] Allerdings gibt es dennoch weitere Anmerkungen, die bezüglich der Sicherheit von DSA zu machen sind.

Wie bei ElGamal-Signaturen ist es auch bei DSA von größter Wichtigkeit, daß der Wert k für jede Signatur neu und zufällig ausgewählt wird und geheim gehalten wird. Kennt ein Angreifer k für eine signierte Nachricht oder besitzt er zwei Nachrichten, die mit dem gleichen k signiert wurden, so kann er den privaten Schlüssel x des Signierenden ableiten und unnachweisbar Signaturen fälschen.[317]

A. K. LENSTRA und S. HABER fanden heraus, daß es für bestimmte Moduli p und q wesentlich einfacher ist, die diskreten Logarithmen zu berechnen und damit Signaturen fälschen zu können. Diese Moduli sind jedoch einfach zu erkennen.[318] Um dieses Problem zu umgehen, hat das NIST im Digital Signature Standard auch eine Vorgehensweise zur Generierung von p und q vorgeschlagen, bei der solche schwachen Werte nicht auftreten.[319]

Das größte Problem bei DSA wurde 1993 von GUSTAVUS J. SIMMONS entdeckt. DSA weist einen sogenannten *verdeckten Kanal* auf. Das bedeutet, daß es durch geschickte Parameterwahl möglich ist, verdeckte Nachrichten in eine DSA-Signatur einzubet-

---

[314] Vgl. ebenda, S. 10.
[315] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 485.
[316] Vgl. **Pedersen, Torben P.:** „Signing Contracts and Paying Electronically“, in: Lectures on Data Security – Modern Cryptology in Theory and Practice, **Hrsg.: Damgård, Ivan B.**, Berlin et al. 1999, S. 140 f.
[317] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 492.
[318] Vgl. **Lenstra, A. K., Haber, S.:** Brief an das NIST betreffend DSS, zitiert bei: **Schneier, Bruce**, 1996, a.a.O., S. 488.
[319] Vgl. **National Institute of Standards and Technology**, 2000, a.a.O., S. 8 u. 13 f.

ten, die nur von Eingeweihten wieder extrahiert werden können.[320] Verdeckte Kommunikation ist bei DSA auf verschiedene Weisen möglich. Allen ist gemeinsam, daß der Sender der verdeckten Nachricht Einfluß auf den verwendeten Wert von k nehmen muß. Dabei ist der verdeckte Kanal so gut versteckt, daß selbst jemand, der die verdeckt kommunizierten Nachrichten kennt, deren Existenz nicht nachweisen kann, solange er nicht in die verwendete Prozedur eingeweiht ist.[321] Die größte Gefahr in diesem Zusammenhang ist, daß es einem böswilligen Implementeur eines DSA-Signatursystems möglich ist, den Algorithmus über den verdeckten Kanal pro Signatur bis zu 10 Bits des privaten Schlüssels des Signierenden absolut unnachweisbar verraten zu lassen. Die einzige Möglichkeit solche verdeckten Kommunikationen zu verhindern, ist daß niemand in der Lage sein darf, den benutzten Wert für k in irgendeiner Weise zu beeinflussen. Im Endeffekt kommt es also auf eine vertrauenswürdige Implementation an, die k absolut zufällig auswählt.[322]

Ein weiteres Problem stellt die maximale Moduluslänge von 1024 Bit dar. Dies ist heute noch ausreichend, kann aber in Zukunft problematisch werden.[323]

Folglich kann DSA momentan als praktisch sicher eingestuft werden, wenn

- die Moduluslänge L ausreichend hoch gewählt wird,
- p und q nach einem Verfahren gewählt werden, das keine schwachen Werte erzeugt und
- die konkrete Implementation k zufällig und für jede Signatur neu auswählt und niemand irgendeinen Einfluß auf den Wert von k hat.

6 Verfahren zur Gewährleistung der Verfügbarkeit

6.1 Problematik

Entsprechend der Definition der Verfügbarkeit aus Abschnitt 3.4.4 muß bei einer Kommunikation sichergestellt werden, daß die Daten in zeitlich akzeptablem Rah-

---

[320] Vgl. **Simmons, Gustavus J.:** „Subliminal Communication is Easy Using the DSA“, in: Advances in Cryptology – EUROCRYPT '93 Proceedings, **Hrsg.: Helleseth, Tor**, Berlin et al. 1994, S. 218 f.
[321] Vgl. ebenda, S. 227 ff.
[322] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 493 u. 535 f.
[323] Vgl. ebenda, S. 486.

men dem Empfänger zur Verfügung stehen und daß sie nicht durch Veränderungen unbrauchbar werden.

Was den zeitlichen Aspekt betrifft, so wird für chiffrierte Kommunikation auf die Überlegungen zur Effizienz von Chiffrierverfahren in den Abschnitten 4 und 7.2 verwiesen.

Veränderungen der Daten können durch Fehler auf den Kommunikationsweg, durch Veränderungen vor der Kommunikation oder durch gezielte Manipulationen eintreten. Bei chiffrierter Kommunikation ist eine Veränderung der Daten durch den Chiffrieralgorithmus selber ausgeschlossen, da alle kryptografischen Algorithmen entsprechend der Definition aus Abschnitt 3.1 die Bedingung $D(E(M)) = M$ erfüllen, die dechiffrierte Nachricht also exakt dem ursprünglichen Klartext entspricht.

Veränderungen auf dem Kommunikationsweg werden in der Regel schon durch das verwendete Übertragungsprotokoll abgefangen, indem den übertragenen Datenblöcken Prüfsummen angehängt werden und so Übertragungsfehler erkannt und die Übertragung gegebenenfalls wiederholt werden kann. Im ISO/OSI-Referenzmodell geschieht dies in der Sicherungsschicht.[324] Dies ist so auch optimal gelöst, da die Fehlererkennung und Fehlerbehebung dann der Chiffrierung nachgelagert ist. Das Anhängen von Prüfsummen an den Klartext macht in diesem Zusammenhang wenig sinn, da ein Fehler im Chiffretext durch Störungen auf dem Kommunikationskanal durch die Fehlerausbreitung bei der Dechiffrierung in der Regel nur schlimmer würde.[325]

Für den Einsatz von kryptografischen Verfahren interessant ist folglich der Bereich der Erkennung von Veränderungen der Daten vor der Versendung und der willentlichen Manipulation entweder vor der Versendung oder durch Abfangen und Austauschen der Nachricht. Diese Funktion sollen kryptografische Hash-Funktionen erfüllen, indem sie, wie in Abschnitt 3.5.4.1 erläutert, eine Art Fingerabdruck der Nachricht erstellen, anhand dessen sich nachträgliche Veränderungen nachweisen lassen.

Im folgenden werden die heute wichtigsten kryptografischen Hash-Funktionen vorgestellt. Dabei wird aufgrund der modernen Anforderungen für digitale Signaturen nur auf die Verfahren eingegangen, die einen Hash-Wert von mindestens 160 Bit Länge erzeugen.

---

[324] Vgl. **Stahlknecht, Peter; Hasenkamp, Ulrich**, a.a.O., S. 124 f.
[325] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 226.

6.2 Kryptografische Hash-Funktionen

6.2.1 Secure Hash Algorithm (SHA-1)

6.2.1.1 Funktionsweise von SHA-1

Der Secure Hash Algorithm (SHA-1) wurde vom NIST im Secure Hash Standard zur Benutzung im Digital Signature Standard veröffentlicht. Es handelt sich dabei um eine Variante der weit verbreiteten Hash-Funktion MD-4 von RSA Data Security Inc. SHA-1 erzeugt einen Hash-Wert der Länge 160 Bit.[326]

Zur Berechnung des Hash-Wertes einer Nachricht M wird diese zunächst aufgefüllt, so daß ihre Länge ein Vielfaches von 512 Bits beträgt, indem zunächst Nullbits angehängt werden, bis die Länge der Nachricht exakt 64 Bits unter dem nächsten Vielfachen 512 Bits liegt, und zuletzt die ursprüngliche Länge der Nachricht als 64-Bit-Zahl angehängt wird. Die aufgefüllte Nachricht wird dann in n Blocks zu 512 Bit, als M_i bezeichnet, aufgeteilt.[327]

Zur Berechnung des 160-Bit-Hash-Wertes werden fünf Akkumulator-Variablen A, B, C, D, E zu je 32 Bit benötigt. Diese werden zuvor auf folgende Werte (in Hexadezimaldarstellung) initialisiert:[328]

$$A = 67452301,$$
$$B = \text{EFCDAB89},$$
$$C = \text{98BADCFE},$$
$$D = 10325476,$$
$$E = \text{C3D2E1F0}.$$

Nun wird der erste Nachrichtenblock verarbeitet. Dabei wird eine Schleife aus 80 Operationen (Laufindex t = 0,1, ..., 79) durchlaufen, wobei in vier Abschnitten je 20 Operationen eine von drei nichtlinearen Funktionen verwenden. Diese Funktionen sind wie folgt definiert:[329]

---

[326] Vgl. **National Institute of Standards and Technology:** FIPS-PUB 180-1, Secure Hash Standard (SHS), Online im Internet, http://csrc.nist.gov/publications/fips/fip180-1/fip180-1.pdf, Stand: 17.04.1995, Abruf: 05.01.2001, S. 1 u. 6.
[327] Vgl. ebenda, S. 8 f.
[328] Vgl. ebenda, S. 10.
[329] Vgl. ebenda, S. 9 f.

$$f_t(x, y, z) = (x \wedge y) \vee ((\neg x) \wedge z), \text{ für } t = 0 \text{ bis } 19,$$
$$f_t(x, y, z) = x \oplus y \oplus z, \text{ für } t = 20 \text{ bis } 39 \text{ und } t = 60 \text{ bis } 79,$$
$$f_t(x, y, z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z), \text{ für } t = 40 \text{ bis } 59.$$

Dabei bezeichnet $\wedge$ eine bitweise UND-Operation, $\vee$ eine Bitweise ODER-Operation, $\oplus$ eine bitweise XOR-Operation und $\neg$ die Umkehrung aller Bits.

Es wird bei jeder Operation eine von vier Konstanten K_t verwendet (ebenfalls hexadezimal):[330]

$$K_t = \text{5A827999}, \quad \text{für } t = 0 \text{ bis } 19,$$
$$K_t = \text{6ED9EBA1}, \quad \text{für } t = 20 \text{ bis } 39,$$
$$K_t = \text{8F1BBCDC}, \quad \text{für } t = 40 \text{ bis } 59,$$
$$K_t = \text{CA62C1D6}, \quad \text{für } t = 60 \text{ bis } 79.$$

Der 512-Bit-Nachrichtenblock M_i wird nun in 16 Unterblöcke $M_{i,t}$ zu je 32 Bit aufgeteilt. Daraus werden dann 80 Datenworte W_t (eines für jede Operation) zu je 32 Bit nach folgender Vorschrift erzeugt:[331]

$$W_t = M_{i,t}, \qquad \text{für } t = 0 \text{ bis } 15,$$
$$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \lll 1, \quad \text{für } t = 16 \text{ bis } 79.$$

Dabei bezeichnet $x \lll n$ eine Linksrotation von x um n Bits.

Nun werden fünf Variablen a, b, c, d, e für die Hauptschleife initialisiert mit $a = A$, $b = B$, $c = C$, $d = D$ und $e = E$. Die nun folgende Hauptschleife, die 80 mal durchlaufen wird, wird am einfachsten als Pseudo-Programmcode dargestellt:[332]

FOR t = 0 TO 79 DO
  PUFFER = $(a \lll 5) + f_t(b, c, d) + e + W_t + K_t$;
  $e = d$; $d = c$; $c = b \lll 30$; $b = a$; a = PUFFER;

Nach dieser Schleife werden die Variablen A, B, C, D und E aktualisiert:[333]

$$A = A + a;\ B = B + b;\ C = C + c;\ D = D + d;\ E = E + e;$$

Diese neuen Werte werden als Anfangswerte für die Verarbeitung des nächsten Datenblocks M_i verwendet. Nachdem der letzte Datenblock verarbeitet wurde, entsteht der 160-Bit-Hash-Wert durch Konkatenation der Variablen A, B, C, D und E.[334]

---

[330] Vgl. ebenda, S. 9.
[331] Vgl. ebenda, S. 10.
[332] Vgl. ebenda, S. 10.
[333] Vgl. ebenda, S. 10.

6.2.1.2 Sicherheit von SHA-1

Es gab einige kryptoanalytische Ansätze gegen die Hash-Funktion MD-4, die für SHA-1, wie oben erwähnt, als Vorlage gedient hat. Auf SHA-1 sind diese Angriffsansätze jedoch nicht mehr anwendbar.[335] Dem Autor war es trotz intensiver Recherche in Fachjournalen, Konferenzpapieren und dem Internet nicht möglich, einen publizierten Angriffsansatz gegen SHA-1 zu finden. Es wird daher davon ausgegangen, daß bis dato keine kryptoanalytischen Angriffe gegen SHA-1 bekannt sind.

Die Hash-Wert-Länge von 160 Bit macht SHA-1 sehr resistent gegenüber Geburtstags-Angriffen.[336]

6.2.2 RIPEMD-160

6.2.2.1 Funktionsweise von RIPEMD-160

Als Reaktion auf die Publikation von Angriffsansätzen gegen MD-4 wurde 1992 im Rahmen des RACE Integrity Primitives Evaluation (RIPE) Programms des EU-Projektes Research and Development in Advanced Communication Technologies in Europe (RACE) eine modifizierte Version von MD-4 vorgeschlagen, die diesen Angriffsansätzen widerstehen sollte. Diese Hash-Funktion wurde RIPEMD genannt.[337] Allerdings fand HANS DOBBERTIN 1995 eine Angriffsmöglichkeit auf eine reduzierte Version von RIPEMD und wies nach, daß diese reduzierte Version nicht kollisionsfrei ist.[338] Daraufhin schlugen 1996 HANS DOBBERTIN, ANTOON BOSSELAERS und BART PRENEEL eine weitere Modifikation vor, die gegen den Angriff von DOBBERTIN resistent ist und deren Hash-Wert-Länge von 128 Bit auf 160 Bit erweitert wurde. Diese Modifizierte Hash-Funktion wurde RIPEMD-160 genannt.[339]

Die Funktionsweise von RIPEMD-160 ist deutlich komplexer, als bei SHA-1. Eine genaue Beschreibung würde den Rahmen dieses Abschnitts sprengen, deswegen sollen hier nur kurz die wichtigsten Unterschiede zu SHA-1 dargelegt werden.

---

334 Vgl. ebenda, S. 10.
335 Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 436 u. 444 f.
336 Vgl. ebenda, S. 445.
337 Vgl. **Dobbertin, Hans:** „RIPEMD with Two-Round Compress Function Is Not Collision-Free“, in: Journal of Cryptology, 10. Jg. (1997), Heft 1, S. 51.
338 Vgl. ebenda, S. 52 f.
339 Vgl. **Dobbertin, Hans; Bosselaers, Antoon; Preneel, Bart:** „RIPEMD-160: A Strengthened Version of RIPEMD“, in: Fast Software Encryption – Third International Workshop Proceedings, **Hrsg.: Gollmann, Dieter**, Berlin et al. 1996, S. 73 f.

Auch RIPEMD-160 benutzt eine Hauptschleife aus 80 Operationen, allerdings unterteilt in 5 Gruppen zu 16 Operationen, so daß 5 nichtlineare Funktionen verwendet werden. Zusätzlich finden Permutationen der Datenworte nach komplexen Tabellen statt und die Anzahl der Linksrotationen ist für jeden Durchlauf der Schleife verschieden. Die Schleife läuft parallel in zwei Instanzen ab, die sich durch verschiedene Anfangswerte, verschiedene Konstanten (folglich gibt es insgesamt 10 Konstanten), umgekehrte Reihenfolge der nichtlinearen Funktionen, unterschiedliche Permutationen der Datenworte und unterschiedliche Bitrotationen unterscheiden. Am Ende der Verarbeitung jedes Datenblocks werden die Ergebnisse beider Instanzen der Hauptschleife zu den fünf Akkumulatoren addiert.[340]

Durch diese erhöhte Komplexität liegen Implementationen von RIPEMD-160 nur bei etwa 86 % des Datendurchsatzes von Implementationen von SHA-1.[341]

6.2.2.2 Sicherheit von RIPEMD-160

Durch die komplexe Struktur ist RIPEMD-160 sehr resistent gegenüber kryptoanalytischen Angriffen.[342] Die Sicherheit von RIPEMD-160 wird im Allgemeinen als äquivalent zu SHA-1 angesehen.[343] Auch für RIPEMD-160 sind bis dato keine Resultate von Angriffsversuchen publiziert worden.

7 Auswahl und Implementation geeigneter Verfahren

7.1 Problematik

Bei der Auswahl eines kryptografischen Verfahrens spielt nicht nur das gebotene Sicherheitsniveau eine Rolle, sondern auch Kriterien wie Effizienz und Eignung für die konkrete Aufgabe. Diese Kriterien beeinflussen die Leistungsfähigkeit der Verfahren im praktischen Einsatz direkt und in erheblichem Maße.

Im folgenden werden die in den vorhergehenden Kapiteln vorgestellten Verfahren einander zusammenfassend gegenübergestellt und anhand der relevanten Kriterien auf ihre Leistungsfähigkeit hin beurteilt. Es werden Auswahlhinweise, sowie wichti-

---

[340] Vgl. ebenda, S. 74 f. u. 80 f.
[341] Vgl. ebenda, S. 78.
[342] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 445.
[343] Vgl. **Menenzes, Alfred J.; van Oorschot, Paul C.; Vanstone, Scott A.:** Handbook of Applied Cryptography, Boca Raton et al. 1997, S. 349.

ge Hinweise für die konkrete Implementation kryptografischer Systeme gegeben und Problembereiche aufgezeigt.

7.2 Verfahren zur Gewährleistung der Vertraulichkeit

Von den vorgestellten symmetrischen Algorithmen DES, IDEA und Blowfish schneidet in punkto Sicherheit DES als mittlerweile veralteter Algorithmus am schlechtesten ab. Es gibt zu viele Angriffspunkte für moderne kryptoanalytische Verfahren, um DES noch guten Gewissens empfehlen zu können.

Das NIST befindet sich daher momentan im Prozeß der Entwicklung eines neuen Standards, der Advanced Encryption Standard (AES) genannt wird und Mitte 2001 in Kraft treten soll. Dazu wurden 1997 Kryptografen aufgerufen, geeignete Algorithmen vorzuschlagen. Die eingereichten Algorithmen wurden seither analysiert und bis 1999 auf fünf Finalisten reduziert: MARS, RC6, Rijdnael, Serpent und Twofish. Im Oktober 2000 wurde die Entscheidung veröffentlicht: Rijndael wurde als Algorithmus für den neuen AES ausgewählt.[344] Rijndael ist, wie alle der Finalisten ein äußerst interessanter Algorithmus mit auf den ersten Blick hohem Sicherheitsniveau, allerdings sind alle fünf Algorithmen zu jung, um hier beurteilt werden zu können. Es bleibt abzuwarten, was die nächsten Jahre der Kryptoanalyse dieser Verfahren ergeben werden. Auf jeden Fall sollte die Entwicklung in diesem Bereich aufmerksam verfolgt werden.

Für die Zwischenzeit wurde vom NIST mit TripleDES versucht, dem Problem der zu geringen Schlüssellänge von DES durch Mehrfachchiffrierung zu begegnen. Dadurch lassen sich jedoch die Erfolge von Verfahren wie der linearen Kryptoanalyse nicht neutralisieren. Und auch der Sicherheitsgewinn durch Verwendung mehrerer unabhängiger Schlüssel ist deutlich geringer, als man augenscheinlich annehmen würde. Daher kann auch TripleDES nicht uneingeschränkt empfohlen werden.

IDEA und Blowfish sind wesentlich jüngere Verfahren und basieren damit auf neueren Erkenntnissen in Kryptografie und Kryptoanalyse. Die Schlüssellänge ist kein Problem mehr, und die Resistenz gegenüber den modernen publizierten kryptoanalytischen Verfahren ist, wie gezeigt wurde, deutlich größer. Obwohl für eine endgülti-

---

[344] Vgl. **National Institute of Standards and Technology:** Report on the Development of the Advanced Encryption Standard (AES),
Online im Internet, http://csrc.nist.gov/encryption/aes/round2/r2report.pdf,
Stand: 02.10.2000, Abruf: 16.01.2001, S. 1 u. 7.

ge Aussage vor allem zu Blowfish noch mehr Analyseergebnisse erforderlich sind, scheint es dennoch angemessen, die Sicherheit beider Verfahren deutlich über DES und TripleDES anzusiedeln, sofern die in den Abschnitten 4.1.2.2 und 4.1.3.2 jeweils herausgearbeiteten Bedingungen für die Sicherheit der Verfahren beachtet werden. Dabei kann die Sicherheit von IDEA und Blowfish bei gleicher Schlüssellänge in etwa als äquivalent angesehen werden, ein Vorteil von Blowfish liegt jedoch in der variablen Schlüssellänge, die das konkrete Sicherheitsniveau variabel gestalten läßt.

Auch was die Effizienz betrifft, ist DES am niedrigsten einzustufen, da IDEA und Blowfish auf der heutigen Hardware beide deutlich schneller arbeiten, vor allem da sie ohne aufwendige Bit-Permutationen auskommen. Bei TripleDES wird die zusätzliche Sicherheit durch die dreifache Ausführungszeit im Vergleich zum einfachen DES teuer erkauft, hier sind IDEA und Blowfish bei höherem Sicherheitsniveau um ein vielfaches schneller. Ein Nachteil von Blowfish in Sachen Effizienz ist die aufwendige Schlüsselexpansion, die vor der Chiffrierung oder Dechiffrierung erfolgen muß. In dieser Hinsicht ist IDEA effizienter. Allerdings verwendet Blowfish deutlich weniger Modulo-Operationen als IDEA, was nach der Schlüsselexpansion wiederum einen Geschwindigkeitsvorteil bedeutet. Dementsprechend ist IDEA bei Anwendungen, in denen kurze Nachrichten mit ständig wechselnden Schlüsseln chiffriert und dechiffriert werden müssen, effizienter, während Blowfish für die Verschlüsselung umfangreicher Datenmengen in längeren Sitzungen, also bei konstantem Schlüssel, Vorteile hat.

Die vorgestellten asymmetrischen Algorithmen RSA und ElGamal sind beide mathematisch wohlfundiert und bieten bei Beachtung der Hinweise aus den Abschnitten 4.2.1.2 und 4.2.2.2 vergleichbare Sicherheit. Von der rechnerischen Komplexität her liegt ElGamal, wie unter Abschnitt 4.2.2.1 erwähnt, etwa um den Faktor 10 höher, als RSA, so daß sich hier ein deutlicher Geschwindigkeitsnachteil gegenüber RSA ergibt. Auch wird die Effizienz von ElGamal durch die erhebliche Nachrichtenexpansion negativ beeinflußt, hier ist der Chiffretext etwa doppelt so lang, wie der Klartext, was bei RSA nicht der Fall ist.

Der erhebliche Geschwindigkeitsnachteil von asymmetrischen Algorithmen gegenüber symmetrischen (DES ist etwa um den Faktor 1.000 schneller, als RSA) macht sie, wie auch schon unter Abschnitt 3.5.2.1 erwähnt, unbrauchbar für die Chiffrierung größerer Datenmengen. Um dennoch in den Genuß des wesentlich einfacheren Schlüsselmanagements der asymmetrischen Algorithmen zu kommen, wird hier empfohlen, ein Hybridverfahren einzusetzen. Als asymmetrischer Algorithmus zur Übertragung der Sitzungsschlüssel wird aufgrund der höheren Effizienz RSA empfohlen. Als symmetrischer Algorithmus zur Chiffrierung der eigentlichen Daten können, abhängig vom Einsatzbereich, IDEA oder Blowfish empfohlen werden.

Die verwendete Schlüssel- (falls variabel) bzw. Moduluslänge sollte in Abhängigkeit von Geheimhaltungszeitraum und potentiellen Angreifern gemäß den Tabellen 1 bis 3 des Anhangs ausgewählt werden. So dürfte bei korrekter Anwendung der Verfahren jedes beliebige Sicherheitsniveau realisierbar sein.

7.3 Verfahren zur Gewährleistung der Verbindlichkeit

Von der Sicherheit her ist an den vorgestellten Signatur-Algorithmen RSA, ElGamal und DSA prinzipiell nichts auszusetzen, sofern die entsprechenden Hinweise für die Implementationen beachtet werden. Nur im Falle von DSA sind hier zwei Einschränkungen angebracht. Bezüglich der Schlüssellänge ist fraglich, inwieweit ein Modulus von höchstens 1024 Bit zukunftsfähig sein wird, besonders unter Berücksichtigung der ständig steigenden Rechenleistungen moderner Computersysteme. Bezüglich des verdeckten Kanals ist anzumerken, daß es sich aufgrund der Unnachweisbarkeit einer verdeckten Kommunikation als schwierig herausstellen kann, eine vertrauenswürdige Implementation zu identifizieren. Wirkliches Vertrauen kann hier nur geschaffen werden, wenn man den Implementeur des Systems kennt und Einsicht in die Entwicklung des Systems nehmen kann.

In puncto Effizienz liegt RSA deutlich in Führung, da dieser Algorithmus den geringsten Rechenaufwand erfordert und keine Nachrichtenexpansion aufweist, während bei ElGamal und DSA die Signatur aus zwei Werten besteht.

RSA ist außerdem der weltweite de facto Standard, der schon in den achtziger Jahren von den meisten großen Computer- und Softwarefirmen (darunter unter anderem Apple, DEC, IBM, Lotus, Microsoft, Novell und Sun) adoptiert wurde und im ISO 9796 Standard festgeschrieben ist.[345] Diese Tatsache, in Verbindung mit der ungeklärten patentrechtlichen Situation von DSA und dem kürzlichen Auslaufen des Patentschutzes für RSA, legt die Nutzung von RSA für digitale Signaturen nahe.

[345] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 484 ff.

Es wird daher an dieser Stelle empfohlen, für digitale Signaturen RSA unter Berücksichtigung der Bedingungen für die Sicherheit des Algorithmus aus den Abschnitten 4.2.1.2 und 5.3.1.2 und unter Verwendung des in Abschnitt 5.2 vorgestellten Protokolls in Verbindung mit einer sicheren kollisionsresistenten Hash-Funktion zu verwenden. Zur Auswahl einer geeigneten Hash-Funktion wird auf die Abschnitte 6.2 und 7.4 verwiesen.

Bei korrekter Anwendung dieser Verfahren wird ein Grad der Fälschungssicherheit erreicht, der nach der Einschätzung des Verfassers die Sicherheit der handschriftlichen Unterschrift auf physikalischen Dokumenten noch übersteigt. Voraussetzung dafür ist aber die Beachtung aller Sicherheitshinweise und die zukunftsorientierte Auswahl von Schlüssellängen, um langfristige Sicherheit zu gewährleisten.

7.4 Verfahren zur Gewährleistung der Verfügbarkeit

Was die Sicherheit der beiden vorgestellten Hash-Funktionen SHA-1 und RIPEMD-160 anbetrifft, so sind diese in etwa gleichwertig. Gegen keines der beiden Verfahren sind bislang kryptoanalytische Angriffe bekannt, das Design fußt auf den gleichen Prinzipien und die erzeugten Hash-Werte haben die gleiche Länge. Dabei ist die Funktionsweise von RIPEMD-160 komplexer, als bei SHA-1. Dieser Umstand kann sich positiv auf die Resistenz des Algorithmus gegenüber Angriffen auswirken, das ist aber nicht zwingend der Fall. Genaueres muß sich hier in Zukunft noch zeigen. Beide Algorithmen sind relativ neu und es kann daher nur empfohlen werden, die zukünftige Entwicklung in diesem Bereich nicht aus dem Auge zu verlieren.

Was die Effizienz betrifft, so hat SHA-1 einen leichten Vorteil gegenüber dem komplexeren RIPEMD-160, was sich besonders bei großen Datenmengen bemerkbar machen wird.

Eine abschließende eindeutige Empfehlung kann nicht gegeben werden, da die beiden Verfahren zu dicht beieinander liegen. Bei knapp bemessener Rechenzeit oder großen Datenmengen könnte man aus Geschwindigkeitsgründen SHA-1 bevorzugen, aber der Performance-Unterschied ist auch nicht allzu groß. Was die Sicherheit betrifft, ist keine Wertung möglich, aber vielleicht wird ein Durchbruch in der Kryptoanalyse diese Frage zugunsten eines der beiden Algorithmen entscheiden.

7.5 Zusätzliche Aspekte

Es sind bei der Implementation eines kryptografischen Sicherheitssystems neben der Suche nach dem sichersten Algorithmus auch weitere Aspekte zu berücksichtigen, von denen das problemadäquate Funktionieren des Systems abhängt.

Zunächst sollte der Aspekt der Angemessenheit der Verfahren beachtet werden. Die verwendeten Algorithmen, Protokolle, Kommunikationssysteme etc. müssen für den konkreten Einzelfall individuell ausgewählt werden, und zwar unter den Gesichtspunkten der potentiellen Gegner, des Geheimhaltungszeitraumes, des Wertes der Information für die Gegner und der Kosten des Sicherheitssystems. Dauert es länger, das zur Chiffrierung von Nachrichten verwendete Kryptosystem zu brechen, als diese Nachrichten geheim gehalten werden müssen, so ist das in der Regel ausreichend. Kostet es den Angreifer mehr, das Kryptosystem zu brechen, als die Daten für ihn wert sind, so dürfte dies auch ausreichen.[346] Dabei ist zu beachten, daß unterschiedlich starke Gegner unterschiedliche Ressourcen haben und dementsprechend die Faktoren des Zeitaufwandes und der Kosten des Gegners von Gegner zu Gegner verschieden sein werden. Letztendlich macht aber die Chiffrierung von Nachrichten aus betriebswirtschaftlicher Sicht nur Sinn, wenn die Kosten des Einsatzes von Kryptografie geringer sind, als die Kosten, die bei Bekanntwerden der betreffenden Nachrichten entstehen.

Die Betrachtung der einzelnen Algorithmen in dieser Arbeit hat gezeigt, daß die bloße Benutzung sicherer Algorithmen keine ausreichende Sicherheit geben kann, da es bei der Implementation auch um Details geht. Werden bei der Implementation kleine Fehler begangen, wie beispielsweise die Vergabe eines gemeinsamen Modulus an alle Systembenutzer bei Verwendung des Algorithmus RSA, so kann ein Kryptoanalytiker diese Ausnutzen und das System brechen. Es ist also äußerst wichtig, daß die zu den hier vorgestellten Algorithmen gegebenen Sicherheitshinweise beachtet werden, und auch keine reduzierten oder unqualifiziert modifizierten Versionen der Algorithmen implementiert werden.

Ein weiterer wichtiger Aspekt ist, daß ein Sicherheitssystem nur maximal so stark sein kann, wie das verwendete Kryptosystem. Auch die Verwendung der besten Algorithmen nützt nichts, wenn es möglich ist, die Algorithmen zu umgehen. Wird z.B. für die Generierung der Sitzungsschlüssel in einem Hybridverfahren ein schlechter

---

[346] Vgl. **Schneier, Bruce**, 1996, a.a.O., S. 8.

Zufallszahlengenerator verwendet, dessen Verhalten vorhersagbar ist, so wird ein Kryptoanalytiker durch Ausnutzung dieser Tatsache das System viel leichter brechen können, als durch aufwendige Angriffe auf den Algorithmus selber. Können durch Unsicherheiten in anderen Bereichen, wie z.B. der Zugriffskontrolle, Schlüssel extrahiert werden, so wird der Angriff dort passieren. Es darf auch nicht die Möglichkeit der Bedrohung von innen unterschätzt werden. Mitarbeiter könnten Schlüssel verraten oder ihre Zugriffsrechte anderweitig mißbrauchen. Es ist also bei weitem nicht genug, RSA oder IDEA oder Blowfish zu verwenden. Das gesamte System muß technisch und organisatorisch sicher sein. Es muß dabei jede Eventualität beachtet werden, denn leider reicht einem versierten Angreifer eine Sicherheitslücke in aller Regel aus, um ein Sicherheitssystem zu umgehen.[347]

8 Schlußbetrachtung

Das Ziel dieser Arbeit war es, kryptografische Verfahren auf ihre Leistungsfähigkeit hin zu beurteilen. Dabei standen die Kriterien der Vertraulichkeit, der Verbindlichkeit und der Verfügbarkeit im Vordergrund. Die grundlegende Frage, die zu beantworten war, ist demnach, ob kryptografische Verfahren in der Lage sind, die genannten Kriterien zu gewährleisten, und wenn ja, in welchem Ausmaß und unter welchen Bedingungen.

Dazu wurde zunächst anhand von einigen aktuellen Beispielen, wie Electronic Procurement oder Supply Chain Management der Bedarf für kryptografische Verfahren in der modernen betrieblichen Umgebung ermittelt. Das Ergebnis dieser Untersuchung ist ein dringender und steigender Bedarf an kryptografischen Verfahren zur Gewährleistung von Transaktionssicherheit und Vertrauen im elektronischen Medium, ohne die neue, vielversprechende Ansätze wie elektronische Märkte oder Supply Chain Management nicht funktionieren können.

Daraufhin wurden die elementaren kryptografischen Grundzusammenhänge erläutert, die die Definition und die Sicherheit kryptografischer Verfahren betreffen, Anforderungen an kryptografische Verfahren aufgestellt und Beurteilungskriterien entwickelt, eine Typisierung von kryptografischen Verfahren vorgestellt und die einzelnen

[347] Vgl. **Schneier, Bruce:** Why Cryptography is Harder than It Looks, Online im Internet, http://www.counterpane.com/whycrypto.pdf, Stand: 1997, Abruf: 16.01.2001, S. 2 ff. und **Anderson, Ross J.**: „Why Cryptosystems Fail“, in: Communications of the ACM, 37. Jg. (1994), Heft 11, S. 36 ff.

Verfahrenstypen ihren Einsatzgebieten zugeordnet, sowie auf die Rolle von kryptografischen Verfahren in Protokollen eingegangen.

Auf Basis dieser Ausführungen war es möglich, diverse konkrete kryptografische Algorithmen den Kriterien Vertraulichkeit, Verbindlichkeit und Verfügbarkeit zuzuordnen und auf ihre Leistungsfähigkeit hin zu untersuchen.

Im Bereich der Gewährleistung der Vertraulichkeit wurden im einzelnen die symmetrischen Algorithmen DES, IDEA und Blowfish, sowie die asymmetrischen Algorithmen RSA und ElGamal vorgestellt und auf das von ihnen gebotene Sicherheitsniveau hin untersucht. Dabei konnten potentielle Schwächen aufgezeigt werden und daraus Bedingungen für die Sicherheit entsprechender Implementationen abgeleitet werden.

Bezüglich der Gewährleistung der Verbindlichkeit wurde zunächst auf die rechtliche Problematik von Geschäftsbeziehungen über das elektronische Medium eingegangen und das Konzept der digitalen Unterschrift als Lösungsansatz für die erläuterte Problematik vorgestellt. Daraufhin wurden mit RSA, ElGamal und DSA drei konkrete Algorithmen zur Erzeugung digitaler Signaturen vorgestellt und auf ihre Sicherheit hin untersucht. Auch hier war es möglich, potentielle Schwächen aufzuzeigen und daraus Bedingungen für die Sicherheit entsprechender Signaturen abzuleiten.

Es wurde dann die Rolle der Kryptografie bei der Gewährleistung der Verfügbarkeit erläutert und als wichtigste Verfahren in diesem Bereich die beiden Hash-Funktionen SHA-1 und RIPEMD-160 vorgestellt.

Abschließend wurden alle vorgestellten Verfahren einander anhand von Sicherheit, Effizienz und Eignung für konkrete Anwendungen gegenübergestellt. Es konnten dabei sinnvolle Empfehlungen bezüglich des Einsatzes gegeben werden. Außerdem wurden Kriterien für die Auswahl von kryptografischen Verfahren herausgearbeitet und wichtige Aspekte der Implementation und des organisatorischen Umfeldes von kryptografischen Verfahren erläutert.

Zusammenfassend hat diese Untersuchung ergeben, daß kryptografische Verfahren in der Lage sind, die Kriterien Vertraulichkeit, Verbindlichkeit und Verfügbarkeit in objektiv ausreichendem Maße zu gewährleisten, sofern entsprechend geeignete Verfahren ausgewählt und sorgfältig alle gegebenen Hinweise für die Implementation und das organisatorische Umfeld beachtet werden. Kryptografie ist ein sehr mächtiges Werkzeug, kann jedoch nur Sicherheit bieten, wenn sie korrekt implementiert wird und nicht durch anderweitige Sicherheitslücken umgangen werden kann. Darüber hinaus darf nicht vergessen werden, daß Kryptografie und Kryptoanalyse außerordentlich dynamische Wissenschaften sind, und beide ständig neue Erkenntnisse hervorbringen. Dabei kann ein Sicherheitssystem nur zukunftsfähig sein, wenn es an neue Entwicklungen und Verfahren angepaßt werden kann und die Parameter für die verwendeten Verfahren auf langfristige Sicherheit hin ausgewählt werden.

Ich versichere, daß ich die vorstehende Arbeit selbständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Hamburg, den 24. Januar 2001 ....

Björn Buske

Anhang

Tabellen

Datentyp	Geheimhaltungs-zeitraum	Minimale Schlüssellänge
Taktische militärische Daten	Minuten – Stunden	56 – 64 Bit
Produktankündigungen, Mergers, Renditen	Tage – Wochen	64 Bit
Langfristige Geschäftspläne	Jahre	64 Bit
Betriebsgeheimnisse	Jahrzehnte	112 Bit
Rüstungsgeheimnisse	> 40 Jahre	128 Bit
Identitäten von Agenten	> 50 Jahre	128 Bit
Diplomatische Informationen	> 65 Jahre	> 128 Bit
U.S.-Zensus-Daten	100 Jahre	> 128 Bit

Tab. 1: Empfohlene Schlüssellängen von symmetrischen Kryptosystemen für Beispieldaten

Quelle: Schneier, Bruce: Applied Cryptography – Protocols, Algorithms, and Source Code in C, 2nd ed., New York et al. 1996, S. 167.

	potentielle Gegner		
Geheimhaltungs-zeitraum (Jahr)	Individuen	Firmen	Regierungen
2000	1024 Bit	1280 Bit	1536 Bit
2005	1280 Bit	1536 Bit	2048 Bit
2010	1280 Bit	1536 Bit	2048 Bit
2015	1536 Bit	2048 Bit	2048 Bit

Tab. 2: Empfohlene Schlüssellängen von asymmetrischen Kryptosystemen

Quelle: Schneier, Bruce: Applied Cryptography – Protocols, Algorithms, and Source Code in C, 2nd ed., New York et al. 1996, S. 162.

Symmetrische Schlüssellänge	Public-key Schlüssellänge
56 Bit	384 Bit
64 Bit	512 Bit
80 Bit	768 Bit
112 Bit	1792 Bit
128 Bit	2304 Bit

Tab. 3: Symmetrische und asymmetrische Schlüssellängen mit vergleichbarer Sicherheit

Quelle: Schneier, Bruce: Applied Cryptography – Protocols, Algorithms, and Source Code in C, 2nd ed., New York et al. 1996, S. 166.

Abbildungen

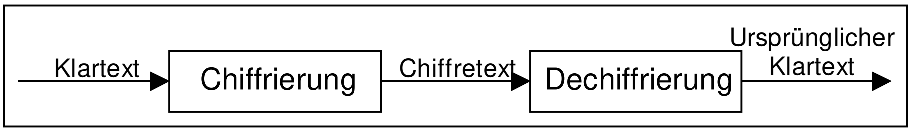

Abb. 1: Chiffrierung und Dechiffrierung von Nachrichten

Quelle: Schneier, Bruce: Applied Cryptography – Protocols, Algorithms, and Source Code in C, 2nd ed., New York et al. 1996, S. 1.

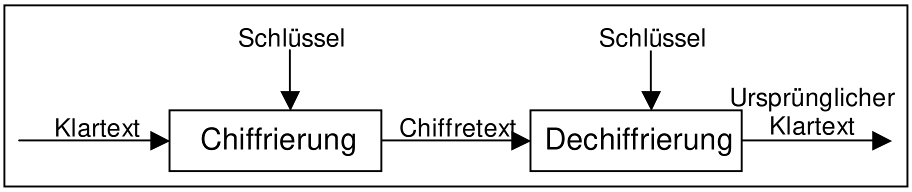

Abb. 2: Chiffrierung und Dechiffrierung mit Schlüssel

Quelle: Schneier, Bruce: Applied Cryptography – Protocols, Algorithms, and Source Code in C, 2nd ed., New York et al. 1996, S. 3.

Anmerkungen

Das One-time Pad

Das *One-time Pad* ist die einzige bekannte Chiffrierungsmethode, die auf kryptoanalytischem Wege nicht gebrochen werden kann. Es wurde 1917 von MAJOR JOSEPH MAUBORGNE und GILBERT VERNAM von AT&T entwickelt. Prinzipiell handelt es sich hierbei um eine extrem lange Folge aus zufällig ausgewählten Schlüsselbuchstaben, in der sich einzelne Abschnitte nicht wiederholen. Sender und Empfänger sind als einzige im Besitz dieser Buchstabenfolge. Der Sender chiffriert mit dem ersten Schlüsselbuchstaben den ersten Buchstaben des Klartextes und so weiter, so daß jeder Schlüsselbuchstabe nur für einen Klartextbuchstaben verwendet wird, indem er $C_i = (M_i + K_i)$ mod 26 bildet (die Buchstaben des Alphabets werden von 1 bis 26 durchnumeriert). Nach der Chiffrierung zerstört der Sender den verwendeten Abschnitt der Folge. Der Empfänger geht zur Dechiffrierung des Chiffretextes analog vor und vernichtet dann ebenfalls denselben Abschnitt des Pads. So wird sichergestellt, daß ein Abschnitt des Pads nur für eine einzige Nachricht verwendet wird.

Klassisch tritt ein One-time Pad in Form eines Einmal-Schlüsselstreifens für Fernschreiber oder als Buchstabenfolge auf einem Papierblock (englisch: *pad*) auf. Benutzte Teile des Schlüsselstreifens, bzw. benutzte Seiten des Blocks werden abgerissen und vernichtet.

So lange ein Angreifer keinen Zugriff auf das zur Chiffrierung verwendete Pad hat, ist die Chiffrierung bedingungslos sicher. Jede denkbare Schlüsselfolge ist gleich wahrscheinlich, somit kann jeder Chiffretext jeden sinnvollen Klartext gleicher Länge darstellen, und der Kryptoanalytiker hat keine Möglichkeit zu bestimmen, welcher Text korrekt ist. So ergibt beispielsweise der Text ONETIMEPAD mit dem Schlüssel TBFRGFARFM den Chiffretext IPKLPSFHGQ. Dechiffriert mit dem Schlüssel POYYAEAAZX, ergibt sich daraus aber SALMONEGGS und mit dem Schlüssel BXFGBMTMXM ergibt sich GREENFLUID. Jedes dieser Ergebnisse ist für den Angreifer gleich wahrscheinlich.

Da die Schlüsselfolge zufällig ist und sich nicht wiederholt, ergibt sich auch eine zufällige Buchstabenfolge als Chiffretext, so daß es unmöglich ist, irgendwelche Informationen über den Klartext daraus zu deduzieren.

Die Nachteile dieses Verfahrens sind jedoch gravierend. Wird kein echtes Zufallsverfahren für die Generierung der Schlüsselfolge verwendet, so kann das Verfahren mathematisch angegriffen werden, indem die statistischen Eigenschaften von Pseudo-Zufallsverfahren ausgenutzt werden. Wird eine Schlüsselfolge mehrmals verwendet, oder überlappen sich die verwendeten Teilfolgen mehrerer Nachrichten, so kann ein Kryptoanalytiker die Klartexte rekonstruieren. Darüber hinaus ist die Speicherung und Verteilung der Schlüssel ungemein problematisch. Jeder Kommunizierende braucht für jeden Kommunikationspartner ein Pad. Davon dürfen nur zwei Kopien existieren, eine beim Sender, die andere beim Empfänger. Die Kopien müssen extrem sicher unter Verschluß gehalten werden. Wenn ein Pad aufgebraucht ist oder kompromittiert wurde, muß ein neues generiert und sicher zugestellt werden. Sender und Empfänger müssen außerdem exakt synchronisiert sein. Liegt der Empfänger auf seinem Pad nur um einen Buchstaben falsch, so kann er die Nachricht nicht entschlüsseln. Wird eine Nachricht auf dem Kommunikationsweg durch Störeinflüsse verfälscht, so ist dies nicht erkennbar und die Entschlüsselung unmöglich.

Aufgrund dieser Nachteile sind One-time Pads nur für kurze Nachrichten auf Kommunikationskanälen mit niedriger Bandbreite und extrem hohen Sicherheitsanforderungen (z.B. Kommunikationen von Geheimagenten) praktikabel.

Anm. 1: Das One-time Pad

Quelle: Vgl. **Schneier, Bruce:** Applied Cryptography – Protocols, Algorithms, and Source Code in C, 2nd ed., New York et al. 1996, S. 15 ff.

Glossar

Algorithmus: Beschreibung einer Prozedur zur Lösung einer Aufgabe.

Algorithmus, asymmetrischer ~: Ein → kryptografischer Algorithmus, bei dem die Schlüssel zur → Chiffrierung und → Dechiffrierung voneinander verschieden sind und sich nicht mit vertretbarem Aufwand voneinander ableiten lassen

Algorithmus, kryptografischer ~: Mathematische Funktion zur → Chiffrierung bzw. → Dechiffrierung von Nachrichten

Algorithmus, Public-key-~: siehe → asymmetrischer Algorithmus

Algorithmus, symmetrischer ~: Ein → kryptografischer Algorithmus, bei dem die Schlüssel zur → Chiffrierung und → Dechiffrierung identisch sind oder zumindest leicht voneinander abgeleitet werden können

Angriff, Adaptive-chosen-plaintext-~: Ein → kryptoanalytischer Angriff, bei dem der → Kryptoanalytiker wiederholt in der Lage ist, von ihm gewählte → Klartexte in ein Kommunikationssystem einzuschleusen, die dazugehörigen → Chiffretexte abzufangen und so die nächsten → Klartexte entsprechend der Ergebnisse der vorherigen abzuändern.

Angriff, Brute-force-~: Ein → kryptoanalytischer Angriff, bei dem der → Kryptoanalytiker den verwendeten → Schlüssel durch vollständiges Durchprobieren des gesamten → Schlüsselraumes zu ermitteln versucht.

Angriff, Chosen-ciphertext-~: Ein → kryptoanalytischer Angriff, bei dem der → Kryptoanalytiker in der Lage ist, von ihm gewählte → Chiffretexte in ein Kommunikationssystem einzuschleusen und die dazugehörigen → dechiffrierten → Klartexte abzufangen.

Angriff, Chosen-plaintext-~: Ein → kryptoanalytischer Angriff, bei dem der → Kryptoanalytiker einmalig in der Lage ist, von ihm gewählte → Klartexte in ein Kommunikationssystem einzuschleusen und die dazugehörigen → Chiffretexte abzufangen.

Angriff, Chosen-key-~: Ein → kryptoanalytischer Angriff, bei dem der → Kryptoanalytiker Informationen über die Beziehungen oder Abhängigkeiten zwischen verschiedenen → Schlüsseln besitzt und dadurch eventuell in der Lage ist, einen → Schlüssel abzuleiten.

Angriff, Ciphertext-only-~: Ein → kryptoanalytischer Angriff, bei dem dem → Kryptoanalytiker lediglich die → Chiffretexte abgefangener Nachrichten zur Verfügung stehen.

Angriff, Known-plaintext-~: Ein → kryptoanalytischer Angriff, bei dem dem → Kryptoanalytiker neben den → Chiffretexten abgefangener Nachrichten auch die dazugehörigen → Klartexte zur Verfügung stehen.

Blockchiffre: Ein → kryptografischer Algorithmus, der den → Klartext in Blöcken fester Länge verarbeitet

Chiffretext: Ergebnis der → Chiffrierung einer Nachricht durch einen → kryptografischen Algorithmus

Chiffrierung: Verschlüsselung einer Nachricht durch einen → kryptografischen Algorithmus

Data Warehouse: Eine zentrale Datenbank, in der alle für das Führungsinformationssystem vorgesehenen Daten aus den Datenbeständen der operativen Anwendungssysteme selektiert und aggregiert zur Verfügung gestellt werden, einschließlich der zugehörigen Instrumente zur Verwaltung, Auswertung und Darstellung der Informationen.

Dechiffrierung: Entschlüsselung einer verschlüsselten Nachricht

Digital Signature Standard: Der amerikanische Standard für → digitale Signaturen.

digitale Signatur: Daten, die an ein elektronisches Dokument angehängt werden oder eine kryptografische Transformation des Dokumentes, wodurch es dem Empfänger ermöglicht wird, Quelle und Integrität der Daten nachzuweisen und das Dokument vor Manipulationen geschützt wird.

E-Commerce: Handel über elektronische Medien.

EDI: Electronic Data Interchange. Elektronischer Datenaustausch zwischen Geschäftspartnern über öffentliche Netze.

EDIFACT: → EDI for Administration, Commerce and Trade. Internationaler Standard zur Übertragung von Handelsdaten.

Einweg-Funktion: Eine mathematische Funktion, deren Umkehrfunktion nicht berechenbar ist

Einweg-Funktion mit Falltür: Ein spezieller Typ von → Einweg-Funktionen, bei dem die Umkehrung rechnerisch undurchführbar ist, es sei denn, man ist im Besitz einer zusätzlichen Information

Electronic Procurement: Elektronisch gestützte Beschaffung mit Hilfe der Technologien des Internet.

Hash-Funktion: Eine mathematische Funktion, die aus einer Bitfolge beliebiger Länge eine Bitfolge fester Länge errechnet

Hash-Funktion, kollisionsresistente ~: Eine → Hash-Funktion, bei der es mindestens schwierig ist, eine → Kollision zu erzeugen.

Hash-Wert: Ergebnis einer → Hash-Funktion

Hybrid-Verfahren: Ein kombiniertes Verfahren aus → symmetrischen und → asymmetrischen Algorithmen, bei dem ein → symmetrischer Algorithmus zur → Chiffrierung der Nachricht eingesetzt wird und mit Hilfe eines → asymmetrischen Algorithmus der → betreffende Sitzungsschlüssel chiffriert ausgetauscht wird

ISO/OSI-Refernzmodell: Standardmodell der ISO zur Verbindung offener Systeme.

IT-System: Informationstechnisches System zur Speicherung, Verarbeitung und Übertragung von Daten

Klartext: Unverschlüsselte Nachricht

Kollision: Es gelingt, zwei Dokumente zu erzeugen, die denselben → Hash-Wert haben

Kryptoanalyse: Wissenschaft von der Entzifferung chiffrierter Nachrichten. Gegenspieler der → Kryptografie

Kryptoanalyse, differentielle ~: → Chosen-plaintext-Angriff, bei dem → Klartextpaare mit einem bestimmten, festen Unterschied chiffriert werden und basierend auf den resultierenden Unterschieden in den → Chiffretexten bestimmten → Schlüsseln Wahrscheinlichkeiten zugeordnet werden. Werden viele → Chiffretextpaare analysiert, so wird irgendwann der gesuchte → Schlüssel zutage treten.

Kryptoanalyse, lineare ~: → Known-plaintext-Angriff, bei dem durch lineare Approximation des → Algorithmus versucht wird, einzelne → Schlüsselbits zu extrahieren.

Kryptografie: Wissenschaft von der Sicherung von Nachrichten gegen unbefugte Einflüsse

Kryptologie: Mathematische Disziplin, bestehend aus → Kryptografie und → Kryptoanalyse

Kryptosystem: Ein → kryptografischer Algorithmus zusammen mit dem gesamten → Schlüsselraum, sowie allen möglichen → Klartexten und den zugehörigen → Chiffretexten

Modulo (mod): Eine mathematische Operation, die den Rest bei der Division zweier ganzer Zahlen ermittelt. Beispiel: 7 mod 3 = 1

Modulus: Rechter Operand einer Modulo-Operation

nonrepudiation: Unmöglichkeit der Nichtanerkennung. Autor bzw. Empfänger einer Nachricht können die Autorenschaft bzw. den Empfang nicht glaubhaft abstreiten

One-time Pad: Ein sehr sicheres, jedoch wenig praktikables Verschlüsselungsschema. Vgl. Anm. 1 des Anhangs.

Protokoll: Vereinbarung über den organisatorischen Ablauf einer Kommunikation bestehend aus einer Reihe von sequentiell durchzuführenden Schritten

Protokoll kryptografisches ~: Ein → Protokoll, bei dem → kryptografische Algorithmen zum Einsatz kommen

Secure Hash Standard: Ein amerikanischer Standard, der die → Hashfunktion SHA-1 für den → Digital Signature Standard definiert.

Sitzungsschlüssel: Ein Schlüssel für einen → kryptografischen Algorithmus, der im Rahmen eines → Hybrid-Verfahrens nur für eine einzige Kommunikation genutzt und dann vernichtet wird

Schlüssel: Eine zusätzliche Information, die ein → kryptografischer Algorithmus zur → Chiffrierung und → Dechiffrierung benötigt

Schlüsselraum: Die Gesamtheit aller zulässigen Werte eines bestimmten → Schlüssels

Stromchiffre: Ein → kryptografischer Algorithmus, der den → Klartext Bit- oder Zeichenweise verarbeitet.

Supply Chain: Alle Aktivitäten, die mit dem Fluß und der Transformation von Gütern vom Rohstoffstadium bis hin zum Endabnehmer verbunden sind, inclusive der zugehörigen Informationsflüsse

Supply Chain Management: Integration der Aktivitäten der Partner in der → Supply Chain durch verbesserte Schnittstellen zwischen den Unternehmen, um gemeinsam einen nachhaltigen Wettbewerbsvorteil zu erreichen

XOR: Exclusiv-oder Verknüpfung zweier Operanden. Die Operanden werden bitweise miteinander verglichen. Das Ergebnis erhält an der entsprechenden Stelle ein Bit mit Wert 1, wenn einer der beiden Operanden an dieser Stelle eine 1 aufweist, nicht jedoch beide. Sind beide betreffenden Bits 0 oder 1, so ist das Ergebnis 0.

Zeitstempel: Eine einem digitalen Dokument hinzugefügte Information eines bestimmten Formats, die das aktuelle Datum und die genaue Tageszeit enthält

Literaturverzeichnis

I. Aufsätze

Alexi, Werner; Chor, Benny; Goldreich, Odad; Schnorr, Claus P.:
„RSA and Rabin Functions: Certain Parts are as Hard as the Whole“,
in: SIAM Journal on Computing,
17. Jg. (1988), Heft 2, S. 194 – 209

Anderson, Ross J.:
„Why Cryptosystems Fail“,
in: Communications of the ACM,
37. Jg. (1994), Heft 11, S. 32 – 40

Anderson, Ross J.; Needham, Roger:
„Robustness Principles for Public Key Protocols“,
in: Advances in Cryptology – CRYPTO '95 Proceedings,
Hrsg.: Coppersmith, Don,
Berlin et al. 1995, S. 236 – 247

Biham, Eli:
„On Modes of Operation“,
in: Fast Software Encryption – Cambridge Security Workshop Proceedings,
Hrsg.: Anderson, Ross J.,
Berlin et al. 1994, S. 116 – 120

Biham, Eli; Biryukov, Alex; Shamir, Adi:
„Miss in the Middle Attacks on IDEA and Khufu“,
in: Fast Software Encryption – 6th International Workshop Proceedings,
Hrsg.: Knudsen, Lars,
Berlin et al. 1999, S. 124 – 138

Biham, Eli; Shamir, Adi:
„Differential Cryptanalysis of DES-like Cryptosystems“,
in: Advances in Cryptology – CRYPTO '90 Proceedings,
Hrsg.: Menenzes, Alfred J.; Vanstone, Scott A.,
Berlin et al. 1991, S. 2 – 21

Biham, Eli; Shamir, Adi:
„Differential Cryptanalysis of the Full 16-round DES“,
in: Advances in Cryptology – CRYPTO '92 Proceedings,
Hrsg.: Brickell, Ernest J.,
Berlin et al. 1993, S. 487 – 496

Bizer, Johann; Miedbrodt, Anja:
„Die digitale Signatur im elektronischen Rechtsverkehr – Deutsches Signaturgesetz und Entwurf der Europäischen Richtlinie“,
in: Handbuch zum Internetrecht: Electronic Commerce – Informations-, Kommunikations- und Mediendienste, **Hrsg.: Kröger, Detlef; Gimmy, Marc A.**,
Berlin et al. 2000, S. 135 – 163

Bogaschewsky, Ronald:
„Electronic Procurement – Neue Wege der Beschaffung“,
in: Elektronischer Einkauf: Erfolgspotentiale, Praxisanwendungen, Sicherheits- und Rechtsfragen,
Hrsg.: Bogaschewsky, Ronald,
Gernsbach 1999, S. 13 – 40

Brickell, Ernest. F.; Moore, Judy H.; Purtill, M. R.:
„Structure in the S-Boxes of the DES“,
in: Advances in Cryptology – CRYPTO '86 Proceedings,
Hrsg.: Odlyzko, A. M.,
Berlin et al. 1987. S. 4 – 7

Campbell, Keith W.; Wiener, Michael J.:
„DES is not a Group“,
in: Advances in Cryptology – CRYPTO '92 Proceedings,
Hrsg.: Brickell, Ernest F.,
Berlin et al. 1993, S. 512 – 517

Coppersmith, Don:
„The Data Encryption Standard (DES) and ist strength against attacks“,
in: IBM Journal of Research and Development,
38. Jg. (1994), Heft 3, S. 243 – 250

Daemen, Joan; Govaerts, René; Vandewalle, Joos:
„Weak Keys for IDEA“,
in: Advances in Cryptology – CRYPTO '93 Proceedings,
Hrsg.: Stinson, Douglas R.,
Berlin et al. 1994, S. 224 – 231

Denning, Dorothy E.:
„Digital Signatures with RSA and Other Public-Key Cryptosystems“,
in: Communications of the ACM,
27. Jg. (1984), Heft 4, S. 388 – 392

Dobbertin, Hans:
„RIPEMD with Two-Round Compress Function Is Not Collision-Free“,
in: Journal of Cryptology,
10. Jg. (1997), Heft 1, S. 51 – 69

Dobbertin, Hans; Bosselaers, Antoon; Preneel, Bart:
„RIPEMD-160: A Strengthened Version of RIPEMD“,
in: Fast Software Encryption – Third International Workshop Proceedings,
Hrsg.: Gollmann, Dieter,
Berlin et al. 1996, S. 71 – 82

ElGamal, Taher:
„A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms“,
in: IEEE Transactions on Information Theory,
31. Jg. (1985), Heft 4, S. 469 – 472

Fuhs, Howard:
„Schutz auf allen Ebenen“,
in: NET, 53. Jg. (1999), Heft 11 – 12, S. 48 – 51

Hastad, Johan:
„On Using RSA with Low Exponent in a Public Key Network“,
in: Advances in Cryptology – CRYPTO '85 Proceedings,
Hrsg.: Williams, Hugh C.,
Berlin et al. 1986, S. 403 – 408

Hawkes, Philip; O’Connor, Luke:
„On Applying Linear Cryptanalysis to IDEA“,
in: Advances in Cryptology – ASIACRYPT '96 Proceedings,
Hrsg.: Kim, Kwangjo; Matsumoto, Tsutomu,
Berlin et al. 1996, S. 105 – 115

Heuser, Ansgar:
„Kryptographie: der Schlüssel zu mehr Datensicherheit in der Informationstechnik“,
in: HMD – Theorie und Praxis der Wirtschaftsinformatik,
33. Jg. (1996), Heft 190, S. 8 – 14

Kelsey, John; Schneier, Bruce; Wagner, David:
„Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER and Triple-DES“,
in: Advances in Cryptology – CRYPTO '96 Proceedings,
Hrsg.: Koblitz, Neal,
Berlin et al. 1996, S. 237 – 251

Knudsen, Lars R.:
„Contemporary Block Ciphers“,
in: Lectures on Data Security – Modern Cryptology in Theory and Practice,
Hrsg.: Damgård, Ivan B.,
Berlin et al. 1999, S. 105 – 126

Kuhlen, Rainer:
„Vertrauenssicherung auf elektronischen Märkten“,
in: WISU, 29. Jg. (2000), Heft 2, S. 220 – 226

Lai, Xuejia; Massey, James L.:
„A Proposal for a New Block Encryption Standard“,
in: Advances in Cryptology – EUROCRYPT '90 Proceedings,
Hrsg.: Dåmgard, Ivan B.,
Berlin et al. 1991, S. 389 – 404
[1991 a]

Lai, Xuejia; Massey, James L.:
„Markov Ciphers and Differential Cryptanalysis“,
in: Advances in Cryptology – EUROCRYPT '91 Proceedings,
Hrsg.: Davies, Donald W.,
Berlin et al. 1991, S. 17 – 38
[1991 b]

Leiberich, Otto:
„Kryptographie – Ist alles unsicher?“,
in: KES, 15. Jg. (1999), Heft 5, S. 14 – 17

Matsui, Mitsuru:
„Linear Cryptanalysis Method for DES Cipher“,
in: Advances in Cryptology – EUROCRYPT '93 Proceedings,
Hrsg.: Helleseth, Tor,
Berlin et al. 1994, S. 386 – 397
[1994 a]

Matsui, Mitsuru:
„The First Experimental Cryptanalysis of the Data Encryption Standard“,
in: Advances in Cryptology – CRYPTO '94 Proceedings,
Hrsg.: Desmedt, Yvo G.,
Berlin et al. 1994, S. 1 – 11
[1994 b]

Meier, Willi:
„On the Security of the IDEA Block Cipher“,
in: Advances in Cryptology – EUROCRYPT '93 Proceedings,
Hrsg.: Helleseth, Tor,
Berlin et al. 1994, S. 371 – 385

Merkle, Ralph C.; Hellman, Martin E.:
„On the Security of Multiple Encryption“,
in: Communications of the ACM,
24. Jg. (1981), Heft 7, S. 465 – 467

Moore, Judy H.:
„Protocol Failures in Cryptosystems“,
in: Proceedings of the IEEE,
76. Jg. (1988), Heft 5, S. 594 – 602

Moore, Judy H.; Simmons, Gustavus J.:
„Cycle Structure of the DES with Weak and Semi-Weak Keys“,
in: Advances in Cryptology – CRYPTO '86 Proceedings,
Hrsg.: Odlyzko, A. M.,
Berlin et al. 1987, S. 9 – 32

Münzenberger, M.:
„Public-Key-Kryptosysteme“,
in: HMD – Theorie und Praxis der Wirtschaftsinformatik,
33. Jg. (1996), Heft 190, S. 31 – 47

Pedersen, Torben P.:
„Signing Contracts and Paying Electronically“,
in: Lectures on Data Security – Modern Cryptology in Theory and Practice,
Hrsg.: Damgård, Ivan B.,
Berlin et al. 1999, S. 134 – 157

Peters, Ralf:
„Technologien des E-Commerce“,
in: WISU, 29. Jg. (2000), Heft 7, S. 961 – 970

Preneel, Bart:
„The State of Cryptographic Hash Functions“,
in: Lectures on Data Security – Modern Cryptology in Theory and Practice,
Hrsg.: Damgård, Ivan B.,
Berlin et al. 1999, S. 158 – 182

Rivest, Ronald L.; Shamir, Adi; Adleman, Leonard:
„A Method for Obtaining Digital Signatures and Public-Key Cryptosystems“,
in: Communications of the ACM,
21. Jg. (1978), Heft 2, S. 120 – 126

Schneier, Bruce:
„Description of a New Variable-Lenght Key, 64-Bit Block Cipher (Blowfish)“,
in: Fast Software Encryption – Cambridge Security Workshop Proceedings,
Hrsg.: Anderson, Ross J.,
Berlin et al. 1994, S. 191 – 204

Schneier, Bruce:
„The Blowfish Encryption Algorithm – One Year Later“,
in: Dr. Dobb’s Journal, 20. Jg. (1995), Heft 9,
Online im Internet,
http://www.counterpane.com/bfdobsoyl.html,
Stand: unbekannt, Abruf: 8.1.2001

Shimoyama, Takeshi; Kaneko, Toshinobu:
„Quadratic Relation of S-box and Its Application to the Linear Attack of Full Round DES",
in: Advances in Cryptology – CRYPTO '98 Proceedings,
Hrsg.: Krawczyk, Hugo,
Berlin et al. 1998, S. 200 – 211

Simmons, Gustavus J.:
„Subliminal Communication is Easy Using the DSA", in:
Advances in Cryptology – EUROCRYPT '93 Proceedings,
Hrsg.: Helleseth, Tor,
Berlin et al. 1994, S. 218 – 232

Vaudenay, Serge:
„On the Weak Keys of Blowfish",
in: Fast Software Encryption – Third International Workshop Proceedings,
Hrsg.: Gollmann, Dieter,
Berlin et al. 1996, S. 27 –32

Werner, Hartmut:
„Supply Chain Management – Partnerschaft zwischen Lieferant und Kunde (I)",
in: WISU, 29. Jg. (2000), Heft 6, S. 813 – 816
[2000 a]

Werner, Hartmut:
„Supply Chain Management – Partnerschaft zwischen Lieferant und Kunde (II)",
in: WISU, 29. Jg. (2000), Heft 7, S. 941 – 945
[2000 b]

Wiener, Michael J.:
„Cryptanalysis of Short RSA Secret Exponents",
in: IEEE Transactions on Information Theory,
36. Jg. (1990), Heft 3, S. 553 – 558

II. Monographien

Buchmann, Johannes:
Einführung in die Kryptographie,
Berlin et al. 1999

Grant, Gail L.:
Understanding Digital Signatures – Establishing Trust over the Internet and other Networks,
New York et al. 1997

Handfield, Robert B.; Nichols, Ernest L., jr.:
Introduction to Supply Chain Management,
London et al. 1999

Kersten, Heinrich:
Einführung in die Computersicherheit,
München, Wien 1991

Menenzes, Alfred J.; van Oorschot, Paul C.; Vanstone, Scott A.:
Handbook of Applied Cryptography,
Boca Raton et al. 1997

Oppliger, Rolf:
IT-Sicherheit – Grundlagen und Umsetzung in der Praxis,
Braunschweig, Wiesbaden 1997

Schneier, Bruce:
Applied Cryptography – Protocols, Algorithms, and Source Code in C, 2nd ed.,
New York et al. 1996

Stahlknecht, Peter; Hasenkamp, Ulrich:
Einführung in die Wirtschaftsinformatik, 8. Aufl.,
Berlin et al. 1997

III. Sonstige Publikationen

Counterpane Systems:
Products That Use Blowfish,
Online im Internet,
http://www.counterpane.com/products.html,
Stand: unbekannt, Abruf: 11.01.2001

National Institute of Standards and Technology:
FIPS-PUB 180-1, Secure Hash Standard (SHS),
Online im Internet,
http://csrc.nist.gov/publications/fips/fip180-1/fip180-1.pdf,
Stand: 17.04.1995, Abruf: 05.01.2001

National Institute of Standards and Technology:
FIPS-PUB 46-3, Data Encryption Standard (DES),
Online im Internet,
http://csrc.nist.gov/publications/fips/fip46-3/fips46-3.pdf,
Stand: 25.10.1999, Abruf: 05.01.2001

National Institute of Standards and Technology:
FIPS-PUB 186-2, Digital Signature Standard (DSS),
Online im Internet,
http://csrc.nist.gov/publications/fips/fip186-2/fips186-2.pdf,
Stand: 27.01.2000, Abruf: 05.01.2001

National Institute of Standards and Technology:
Report on the Development of the Advanced Encryption Standard (AES),
Online im Internet,
http://csrc.nist.gov/encryption/aes/round2/r2report.pdf,
Stand: 02.10.2000, Abruf: 16.01.2001

Regulierungsbehörde für Telekommunikation und Post:
Geeignete Kryptoalgorithmen gemäß § 17 (2) SigV,
Online im Internet,
http://www.regtp.de/imperia/md/content/tech_reg_t/digisign/23.pdf,
Stand 28.12.2000, Abruf 28.12.2000

Schneier, Bruce:
Why Cryptography is Harder than It Looks,
Online im Internet,
http://www.counterpane.com/whycrypto.pdf,
Stand: 1997, Abruf: 16.01.2001