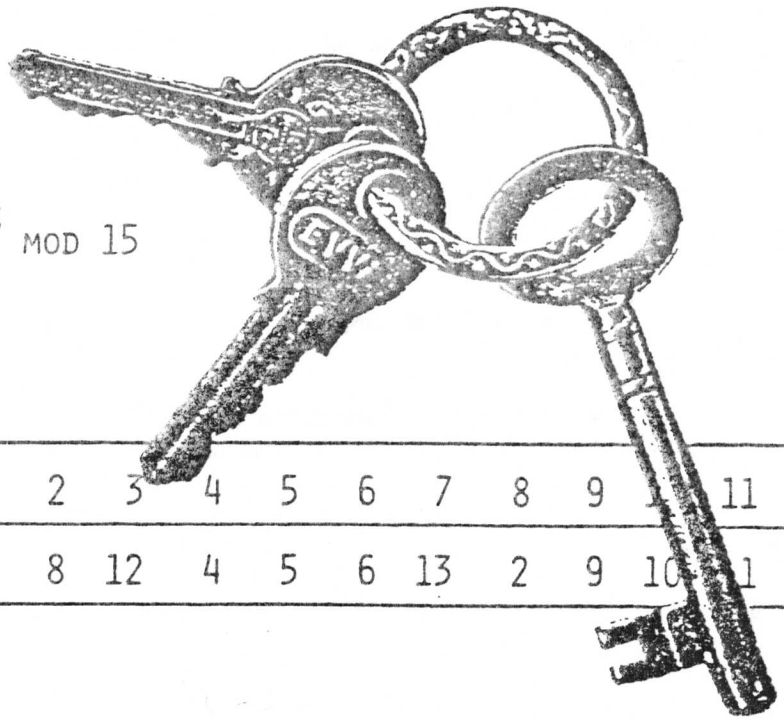
LEHRERFORTBILDUNGSTAGUNG 1984

WINFRIED B. MÜLLER
INSTITUT FÜR MATHEMATIK
UNIVERSITÄT KLAGENFURT

MATHEMATISCHE METHODEN BEI PROBLEMEN DES DATENSCHUTZES

$x \longrightarrow x^3 \bmod 15$



x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$x^3 \bmod 15$	0	1	8	12	4	5	6	13	2	9	10	1	3	7	14

MATHEMATISCHE METHODEN BEI PROBLEMEN DES DATENSCHUTZES

1. EINLEITUNG UND GRUNDLAGEN

Mit dem zunehmenden Einsatz elektronischer Kommunikationssysteme in den verschiedensten Bereichen - z.B. bei elektronischen Nachrichtenübertragungen, im bargeldlosen Zahlungsverkehr oder bei der computergesteuerten Überwachung hochkomplexer Systeme (Industrieanlagen, Kraftwerke, Raketenbahnen) - sind neue Probleme des Datenschutzes entstanden. Informationen müssen vor unbefugtem Zugriff geschützt werden, und die Authentizität empfangener Nachrichten muß überprüfbar sein.

Sicherheitsprobleme im Computer-Zeitalter

(i) Informationsverarbeitung
   a) Datenschutz
   b) Bargeldlose Gesellschaft
   c) Schutz von Software

(ii) Kommunikations- und Nachrichtentechnik
   a) Elektronische Post
   b) Elektronische Geldüberweisungen

(iii) Steuer- und Regelungstechnik
   a) Steuerung von Industrieanlagen
   b) Steuerung von automatischen Fahrzeugen

Eine Möglichkeit, sich gegen den Mißbrauch von Daten zu schützen, besteht in der Verschlüsselung der betreffenden Informationen.

Um eine Nachricht zu verschlüsseln, formt man normalerweise den Klartext mit Hilfe eines "digitalen Alphabets" in eine Zahlenfolge um. Auf diese Zahlenfolge bzw. Abschnitte davon wird dann eine Verschlüsselungsfunktion oder Chiffrierfunktion E_i angewendet, wodurch man den verschlüsselten Text (Chiffrat) erhält. Das Chiffrat wird über einen öffentlichen (unsicheren) Kanal an dem Empfänger übermittelt.

Zu jeder Chiffrierfunktion E_i gibt es eine Entschlüsselungsfunktion oder Dechiffrierfunktion D_i (d.h. die Funktionen E_i müssen injektiv sein und D_i ist die Umkehrfunktion von E_i), welche dem Empfänger erlaubt, das Chiffrat in den Klartext rückzutransformieren.

Damit der rechtmäßige Empfänger weiß, welche Dechiffrierfunktion D_i er auf ein empfangenes Chiffrat anzuwenden hat, muß ihm über einen geheimen (sicheren) Kanal der sogenannte Schlüssel K_i übermittelt werden, das ist eine Information, durch welche die Dechiffrierfunktion D_i eindeutig festgelegt wird.

Nachteile der "klassischen" Chiffrierung

(i) K_i kann bei der Übermittlung in falsche Hände geraten.

(ii) Je mehr Personen K_i kennen, desto schwieriger ist die Geheimhaltung.

(iii) Unter Umständen kann aus der Kenntnis von Nachrichtenteilen und Chiffratteilen der Schlüssel K_i berechnet werden (Gegenmaßnahme: Verschlüsselung mit Zufallsfolgen).

(iv) Kontaktaufnahme von fremden Personen ist nicht möglich, da zuerst der Schlüssel K_i ausgetauscht werden muß.

(v) Authentizität des Absenders ist nicht ohne weiteres überprüfbar.

Modell einer elektronischen Nachrichtenübertragung

Bei der Übertragung elektronischer Nachrichten (siehe Abb.1) führt man üblicherweise zwei Transformationen durch: Man chiffriert die Nachricht um sie geheim zu halten und man codiert die chiffrierte Nachricht (man fügt Prüf- und Korrekturinformationen hinzu), um nach der Übertragung hinzugekommene Fehler oder Störungen erkennen und womöglich auch korrigieren zu können. Die wissenschaftliche Disziplin, welche sich mit der Transformation von Nachrichten zu Geheimhaltungszwecken befaßt heißt Kryptographie. Die Wissenschaft, welche sich mit den Problemen der Fehlererkennung und Fehlerkorrektur bei der Übertragung elektronischer Nachrichten befaßt, nennt man Codierungstheorie.

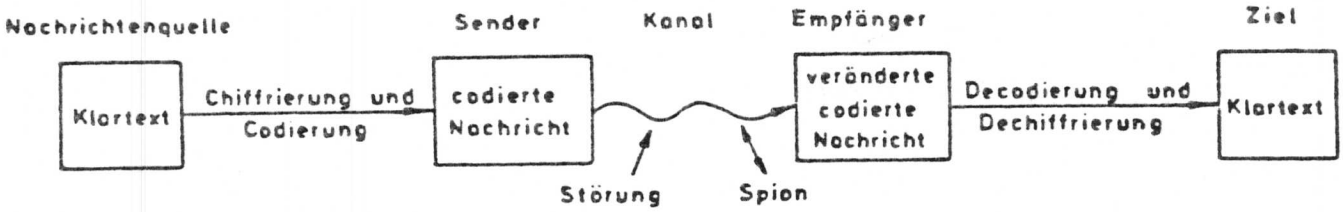


Abb.1

Im folgenden wollen wir uns mit den mathematischen Methoden in der Kryptographie etwas ausführlicher befassen.

2. METHODEN DER KLASSISCHEN CHIFFRIERUNG

Diplomaten und Militärs schützen Daten und Nachrichten bereits seit über 2000 Jahre durch Verschlüsseln. Schon J. Caesar (100 - 44 v.Chr.) wird ein Verschlüsselungsverfahren zugeschrieben, bei welchem jeder Buchstabe des Alphabets durch den Buchstaben des Alphabets ersetzt wird, welcher drei Plätze nach ihm im Alphabet kommt. Damit ergibt sich die in Tabelle 1 dargestellte Permutation des Alphabets.

Klartext:	A	B	C	...	W	X	Y	Z
Chiffrat:	D	E	F	...	Z	A	B	C

Tabelle 1

Auch kompliziertere Verschlüsselungen dieser Art haben jedoch den Nachteil, daß man sie auf Grund der charakteristischen

Häufigkeit der einzelnen Buchstaben, Buchstabenpaare usw. in jeder Sprache leicht entziffern kann, wenn man nur genügend Text kennt.

Die in Tabelle 2 dargestellte Tafel, durch welche den Buchstaben des Alphabets Zahlen zwischen 1 und 55 zugeordnet werden, stammt vom griechischen Schriftsteller Polybius (ca. 150 v.Chr.). Die Polybius-Tafel ist deswegen bemerkenswert, da sie historisch das erste Beispiel für eine digitale Zuordnung ist. Jeder Buchstabe wird durch eine zweiziffrige Zahl ersetzt, wobei die erste Ziffer der Zahl die Zeile angibt, in welcher der Buchstabe in Tabelle 2 steht, und die zweite Ziffer seine Spalte (z.B. M ↔ 32, U ↔ 45).

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I=J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	U	W	X	Y	Z

Klartext: "NICHT"

Chiffrat: "3324132344"

Tabelle 2

Da die Polybius-Methode das Anzeigen der Buchstaben mittels der Finger der rechten und linken Hand ermöglicht, wird diese Methode auch in Gefängnissen und anderen Orten mit erschwertem akustischen Nachrichtenaustausch zur optischen Obermittlung von Informationen verwendet (vgl. Solschenizyn: Archipel Gulag).

Wir wollen nun eine Verschlüsselungsmethode besprechen, welche wegen ihrer "absoluten" Sicherheit bei vielen wichtigen Anwendungen der Kryptographie verwendet wird. Auch beim sogenannten "Roten Telefon", welches Moskau mit Washington verbindet, und welches in Wirklichkeit aus 6 Fernschreibern (derzeit findet eine Umrüstung auf Telekopierer statt) besteht, die bei Bedarf Nachrichten von Moskau nach Washington übermitteln und umgekehrt. Damit niemand Unbefugter diese Gespräche abhören oder verfälschen kann, verschlüsselt man sie. Die dafür verwendete Methode der *Verschlüsselung mittels Ein-Weg-Schablone mit Zufallsziffern* wurde 1926 vom

Amerikaner G.S. Vernam entwickelt und ist das einzige mathematisch nachweislich nicht knackbare Verschlüsselungssystem. Das Verfahren funktioniert folgendermaßen:

Man erzeugt eine lange binäre Zufallsfolge $k = a_1a_2a_3\ldots$, $a_i \in GF(2)$. Theoretisch kann man dies z.B. durch Aufwerfen einer Münze machen. Sender und Empfänger erhalten je eine Kopie von k und halten diese geheim. Will nun der Sender eine Nachricht übermitteln, so transformiert er diese, etwa unter Zuhilfenahme des in Tabelle 3 gegebenen dualen Alphabets in eine Zahlenfolge $m = b_1b_2b_3\ldots$ und addiert m Stelle für Stelle modulo 2 zu k. Damit erhält der Sender die verschlüsselte Nachricht c, welche er übermittelt. Außerdem vernichtet der Sender die von der Folge k verbrauchten Anfangsglieder. Der Empfänger addiert zur Folge c noch einmal die Folge k und erhält damit wieder m, woraus er sich mittels Tabelle 3 die Nachricht rekonstruieren kann. Auch der Empfänger vernichtet die von ihm verbrauchte Teilfolge von k.

Diese Verschlüsselungsmethode hat zwar den Vorteil der absoluten Sicherheit, man muß jedoch auf Vorrat sehr lange binäre Zufallsfolgen erzeugen und diese unter strengster Geheimhaltung austauschen. (Im Falle des roten Telefons geschieht dies abwechselnd über die Botschaften der SU und USA.) Jede Zufallsfolge darf nur einmal verwendet werden.

Viele der heute in Gebrauch befindlichen Verschlüsselungen beruhen auf dem soeben beschriebenen Verfahren. Auch der von den amerikanischen Bundesbehörden für Verschlüsselungen verwendete *Data Encryption Standard* (DES), auf welchen wir noch später zu sprechen kommen, verwendet binäre Zufallsfolgen.

A := 000000	0 := 011011	, := 100101
B := 000001	1 := 011100	. := 100110
C := 000010	2 := 011101	
D := 000011	.	.
E := 000100	.	.
.	.	.
.	.	.
.	.	.
Z := 011001	9 := 100100	.

Tabelle 3

Beispiel: Wir erzeugen durch Münzaufwurf die binäre Zufalls - folge

$$k = 011010001111010111001010 00...$$

Nun wollen wir die Nachricht "NEIN" senden. Unter Zuhilfenahme von Tabelle 3 verwandeln wir die Nachricht in die binäre Folge

$$m = 001101\ 000100\ 001000\ 001101$$

und addieren modulo 2 (symbolisch: $\oplus_2$):

$$\begin{array}{rl} & m : 00110100010000100000 1101 \\ & k : 011010001111010111001010 00... \\ \hline c = m \oplus_2 k & : 010111001011011111000111 \end{array}$$

Nun wird c übermittelt. Der Empfänger bildet $c \oplus_2 k = m \oplus_2 k \oplus_2 k = m$, woraus er sich mittels Tabelle 3 die Nachricht "NEIN" rekonstruiert.

$$\begin{array}{rl} & c : 010111001011011111000111 \\ & k : 011010001111010111001010 00... \\ \hline c \oplus_2 k = m & : 001101|000100|001000|001101 \\ & \quad N \quad E \quad I \quad N \end{array}$$

Nun wollen wir noch kurz auf den schon erwähnten DES-Algorithmus eingehen. Der Data Encryption Standard wurde auf Antrag der US-Regierung vom National Bureau of Standards in den siebziger Jahren in Zusammenarbeit mit IBM entwickelt. Amerikanische Bundesbehörden sind verpflichtet, ihre Daten mit DES zu chiffrieren. Aber auch viele Banken und Industrieunternehmen haben sich bereits DES-Zusatzgeräte zu ihren Computern gekauft.
Der DES-Algorithmus verschlüsselt Datenblöcke von 64 Binärstellen. Der Benutzer erzeugt sich durch Münzwurf acht 7-stellige binäre Zufallsblöcke. An jeden dieser Blöcke wird noch eine Parity-check-Stelle angehängt. Die so erhaltenen 64 Binärstellen sind der "Schlüssel" K für den DES-Algorithmus. DES selbst ist ein relativ komplizierter Algorithmus, der mit Hilfe von K jede Nachricht m in eine Pseudozufallsfolge K(m) verwandelt. Eine offizielle Beschreibung des DES-Algorithmus kann z.B. in Katzan (1977) nachgelesen werden. Im wesentlichen besteht der Algorithmus aus drei Teilen:

Einer fixen anfänglichen Permutation der Komponenten des eingegebenen Nachrichtenvektors; einer komplizierten, von K abhängigen Transformation, bei der die Komponenten in 16 Stufen abwechselnd permutiert und nichtlinear auf sich selbst abgebildet werden; und schließlich einer Schlußpermutation, welche invers zur anfänglichen Permutation ist. Der DES-Algorithmus ist demnach eine Chiffrierung, welche auf die Ein-Weg-Schablone mit Zufallsziffern aufbaut. Das Kernstück der oben erwähnten Transformation bilden acht nichtlineare Funktionen. Geheim ist nur der Schlüssel K. Alle verwendeten Funktionen sind bekannt. Da sie jedoch nach geheimen Prinzipien konstruiert wurden, wird von Gegnern des Verfahrens eingewendet, daß Insider auch ohne Kenntnis des Schlüssels K den DES unter Umständen entziffern könnten. Dieser Verdacht wird unter anderem durch die Beschränkung des Schlüssels auf 64 Stellen begründet.

3. CHIFFRIERSYSTEME MIT ÖFFENTLICHEM SCHLOSSEL

Wie schon erwähnt, spielt die Kryptographie heute nicht nur im militärischen und diplomatischen Bereich eine Rolle, sondern es treten zunehmend auch auf dem öffentlichen und kommerziellen Sektor Probleme auf, für welche die Chiffrierung von Bedeutung ist. Allerdings versagen hierbei vielfach die sogenannten klassischen Chiffriersysteme, zu denen auch die Ein-Weg-Schablone mit Zufallsziffern gehört. Der moderne Welthandelt erfordert einen vertraulichen Nachrichtenaustausch zwischen Handelspartnern, die oft nicht vorher Gelegenheit haben, irgendwelche "Chiffrierschlüssel" auszutauschen. Zu den Problemen des Datenschutzes gehört auch das Problem, wie man elektronische Briefe vertraulich halten und signieren kann (zwei wichtige Eigenschaften der klassichen Briefpost).

1976 hatten die Amerikaner W. Diffie und M. Hellmann (1976) die Idee für eine revolutionäres Chiffrierverfahren, das

man *Public-Key Kryptosystem* (Verschlüsselung mit öffentlich bekanntem Schlüssel) nennt und welches viele der heutigen Anforderungen des Datenschutzes und der Geheimhaltung zu erfüllen scheint.

Ein Public-Key Kryptosystem arbeitet folgendermaßen:

Jede Person X, welche am geheimen Nachrichtenaustausch teilnehmen möchte, gibt eine Verschlüsselungsfunktion E_X bekannt und hält eine Dechiffrierfunktion D_X geheim. E_X und D_X haben dabei die folgenden Eigenschaften:

(1) Ist m eine Nachricht, so gilt $D_X(E_X(m)) = m$.
(2) E_X und D_X sind beide leicht am Computer auszuführen.
(3) Es ist nur unter sehr großem Rechenaufwand möglich (d.h. praktisch unmöglich), D_X aus E_X zu berechnen. Derartige Funktionen E_X heißen Falltürfunktionen.

Will nun A an B eine Nachricht m senden, dann besorgt sich A aus einem öffentlich aufliegenden Verzeichnis (analog einem Telephonbuch) E_B und sendet $E_B(m)$ an B. Nur Person B (eventuell nur sein Computer) kennt D_B und berechnet $D_B(E_B(m))$, was nach (1) den Klartext m ergibt.

Ist die Menge aller Nachrichten M gleich der Menge aller Chiffrate C, und gilt auch noch die Eigenschaft

(4) $E_X(D_X(m)) = m$ für jede Nachricht $m \in M$,

dann kann A Nachrichten auch signieren. Dazu sendet er $E_B(D_A(m))$ an B. Dieser wendet zum Entschlüsseln zunächst seinen geheimen Schlüssel D_B auf die verschlüsselte Nachricht an und danach die dem öffentlichen Verzeichnis entnommene Chiffrierfunktion E_A. Damit bekommt er $E_A(D_B(E_B(D_A(m)))) = m$. Da nur A die Funktion D_A kennt, muß die Nachricht von A kommen, und nur B kann sie lesen, da nur er Zugriff zu D_B hat. (Ist für B nicht von vornherein klar, daß die empfangene Nachricht von A kommt, so sendet A vor der "signierten Nachricht" $E_B(D_A(m))$ eine einfach verschlüsselte Nachricht $E_B(m)$ an B, in der er den Absender mitteilt.)

Nach der Formulierung der Idee eines Public-Key Kryptosystems hat es zunächst fast 2 Jahre gedauert, bis ein konkretes Public-Key System vorgestellt wurde. Auch bis heute kennt man im wesentlichen nur die folgenden Realisierungen von Public-Key Kryptosystemen:

(1) RSA-Schema und Varianten (Rivest, Shamir und Adlemann (1978), Müller und W. Nöbauer (1981), R. Nöbauer (1984))

(2) MH-Schema und Varianten (Merkle und Hellman (1978))

(3) Mc.Eliece-Schema (Mc.Eliece (1978), Sloane (1981))

(4) Massey-Omura Lock (Massey-Omura (1983))

Aufgrund neuerer Arbeiten (Shamir (1983), Ingemarsson (1983), Eier und Lagger (1983)) kann das auf dem sogenannten Untersummenproblem beruhende MH-Schema nicht mehr als sicher gelten.

Die Sicherheit des auf den Goppa Codes basierenden Mc.Eliece-Schemas ist noch nicht eingehend geprüft. Die praktische Verwendung ist wegen sehr großer Schlüssel (die Schlüssel bestehen aus großen Matrizen) schwierig. Das Massey-Omura Lock beruht auf einer neuen Multiplikationsmethode in Körpern der Ordnung 2^n. Für die in der Praxis notwendigen großen Körper (n=500), befindet es sich aber noch im Konstruktionsstadium.

Damit stehen derzeit für den praktischen Einsatz nur das RSA-Verfahren und Varianten in Software und Hardware zur Verfügung. Dieses Kryptosystem basiert auf dem Faktorisierungsproblem großer natürlicher Zahlen. Wir werden uns im nächsten Abschnitt mit dem RSA-Kryptosystem genauer befassen.

Die geringe Anzahl konkreter Public-Key Kryptosysteme steht im Gegensatz zur fundamentalen Bedeutung, welche diesen Systemen in den wenigen Jahren seit ihrer allgemeinen Formulierung heute bei Fragen des Datenschutzes eingeräumt wird und die ihnen keineswegs nur aus theoretischem Interesse, sondern auch und vor allem vom Standpunkt des Anwenders aus betrachtet zukommt.

4. DAS RSA-KRYPTOSYSTEM (Rivest/Shamir/Adleman (1978))

Das RSA-Verschlüsselungssystem basiert auf dem Problem, eine große Zahl zu faktorisieren, das heißt, alle Primzahlen zu finden, durch die sich die Zahl ohne Rest teilen läßt. (Nach dem Hauptsatz der Zahlentheorie läßt sich jede ganze Zahl ungleich Null je nach ihrem Vorzeichen als Produkt von 1 oder -1 und endlich vielen Primzahlen p_i mit $p_i \leq p_{i+1}$ darstellen.)

Beim RSA-Verfahren nutzt man die Tatsache aus, daß es fast keine Zeit kostet, das Produkt zweier großer Primzahlen zu berechnen, daß es aber kein schnelles Verfahren gibt, um eine große Zahl in ihre Primfaktoren zu zerlegen.

Man berechne z.B. das Produkt $127 \cdot 229$ und faktorisiere andererseits 29 083 und vergleiche den Zeitaufwand.

Das Faktorisieren großer Zahlen (man verwendet heute Zahlen mit 500 Stellen) ist auch mit den größten und schnellsten Rechnern ein großes Problem (der Zeitaufwand zum Faktorisieren steigt exponentiell mit der Anzahl der Stellen der Zahl). Tabelle 6 informiert über die Fortschritte beim Faktorisieren großer Zahlen in den letzten Jahren.

Jeder Benützer des RSA-Systems erzeugt mit dem Zufallszahlengenerator eines Computers zwei große Primzahlen p und q und eine weitere Zufallszahl k, die zu (p-1)(q-1) relativ prim sein soll (weil nur unter dieser Bedingung die zu erzeugende Funktion injektiv ist). (Bei der Bestimmung dieser großen Primzahlen tritt das Problem auf, daß es keinen genügend raschen Algorithmus gibt, der mit absoluter Sicherheit feststellen kann, ob es sich bei den gefundenen Zahlen wirklich um Primzahlen handelt oder nicht. Für die Praxis reicht es jedoch, wenn man feststellen kann, ob es sogenannte Pseudoprimzahlen, also mit einer hohen Wahrscheinlichkeit Primzahlen, sind. Dazu wurde 1976 von M.O. Rabin (1980) ein Verfahren entwickelt, auf welches wir aber hier nicht näher eingehen.) Das Produkt $n = pq$ und k werden öffentlich bekanntgegeben, während die Zahlen p und q nur

dem Empfänger der Nachricht bekannt sind. Weder der Absender noch ein Dritter kennen sie und können sie auch nicht innerhalb einer annehmbaren Zeit berechnen, wodurch, wie wir sehen werden, die Geheimhaltung gewährleistet ist. Der Absender einer Nachricht formt diese zunächst z.B. mit dem in Tabelle 4 gegebenen dezimalen Alphabet in eine Kette von Dezimalzahlen um.

A = 01	B = 02	C = 03	D = 04	E = 05
F = 06	G = 07	H = 08	I = 09	J = 10
K = 11	L = 12	M = 13	N = 14	O = 15
P = 16	Q = 17	R = 18	S = 19	T = 20
U = 21	V = 22	W = 23	X = 24	Y = 25
Z = 26				

Tabelle 4

Zum leichteren Verständnis besprechen wir das Verfahren an einem *Beispiel:*

A will die Nachricht "JA" an B übersenden. Von B ist der öffentliche Schlüssel $K_B = (k,n) = (193,7031)$ bekannt, den B aus seinen selbst ermittelten Werten $p = 89$, $q = 79$ und $k = 193$ erhalten hat.

A schreibt die Nachricht m zunächst mit Hilfe von Tabelle 4 als Dezimalzahl und erhält

$$\text{"JA"} = 1001 = m$$

Anschließend zerlegt er diese Zahl in gleich lange Blöcke $m_1, m_2, \ldots$, sodaß die Werte von m_i immer kleiner als n bleiben. Wir können $m_1 = 1001$ wählen. Mit dem öffentlich bekannten Schlüssel (k,n) von B (der Schlüssel ist also ein Paar von Zahlen) wird jeder Block in eine chiffrierte Zahl umgewandelt und zwar durch die Chiffrierfunktion E_B, gegeben durch .

$$c_i = m_i^k \bmod n \,.$$

Also erhalten wir $c_1 = m_1^k \bmod n = 1001^{193} \bmod 7031 = 4494$.

E_B wählt man deshalb als Modulfunktion, weil bei alleiniger Verwendung der Potenzfunktion m_i^k sofort auch die Umkehrfunktion c_i^r ($r = 1/s$) bestimmt werden könnte.

Die Zahlen $c_1, c_2, \ldots$ bilden dann das Chiffrat c. Sie werden über einen öffentlichen Kanal an den Empfänger weitergeleitet. Diesen chiffrierten Text entschlüsselt der Empfänger B mit dem nur ihn bekannten Dechiffrierschlüssel D_B, den er aus den Zahlen k, p und q konstruiert.

Wie man den Dechiffrierschlüssel D_B berechnet, soll in der Folge gezeigt werden.

Dazu benötigen wir zunächst den

Kleinen Satz von FERMAT:

Ist p *eine Primzahl, so gilt:* $a^p \equiv a \bmod p$ *für alle* $a \in \mathbb{N}$.
Ist a *überdies relativ prim zu* p, *also* $ggT(a,p) = 1$, *so gilt:*

$$a^{p-1} \equiv 1 \bmod p.$$

Beweis:

(1) Wir zeigen die Behauptung durch vollständige Induktion nach a.
Für $a = 1$ ist die Behauptung nichtig, da stets $1^p \equiv 1 \bmod p$ gilt.
Sei die Behauptung nun für $a \in \mathbb{N}$ richtig. Dann gilt $a^p \equiv a \bmod p$ oder anders formuliert $p/(a^p-a)$ (p teilt a^p-a).
Nun muß aber auch $(a+1)^p \equiv (a+1) \bmod p$ gelten, denn

$$(a+1)^p - (a+1) = a^p + \binom{p}{1}a^{p-1} + \binom{p}{2}a^{p-2} + \ldots + \binom{p}{p-1}a + 1 - a - 1, \text{ also}$$

$$(a+1)^p - (a+1) = (a^p-a) + \binom{p}{1}a^{p-1} + \binom{p}{2}a^{p-2} + \ldots + \binom{p}{p-1}a.$$

Da für prime p jedoch $p/\binom{p}{i}$ für $i \leq i \leq p-1$ und $p/(a^p-a)$ gilt, ist jeder der Summanden in obiger Gleichung auf der rechten Seite durch p teilbar, das heißt aber $p/((a+1)^p - (a+1))$ oder $(a+1)^p \equiv (a+1) \bmod p$.
Somit ist die erste Behauptung für alle $a \in N$ bewiesen.

(2) Sei nun ggT(a,p) = 1, dann folgt aus der ersten Behauptung $a^p \equiv a \bmod p$ oder $p/a(a^{p-1} - 1)$. Da a und p teilerfremd sind, muß $p/(a^{p-1} - 1)$ gelten und damit $a^{p-1} \equiv 1 \bmod p$ sein.

Ist n das Produkt zweier Primzahlen, so wie es beim RSA-Schema der Fall ist, dann läßt sich obiger Satz folgendermaßen verallgemeinern:

Satz von Euler-Fermat: *Ist* $n = pq$ *mit* p,q *prim und* $p \neq q$, *dann gilt* $a^{1+t(p-1)(q-1)} \equiv a \bmod pq$ *mit beliebigen* $t \in \mathbb{Z}$.

Beweis: Fall 1: Ist ggT(a,n) = 1, dann gilt nach dem kleinen Fermat'schen Satz $a^{p-1} \equiv 1 \bmod p$ und $a^{q-1} \equiv 1 \bmod q$.

Potenziert, ergibt dies

$a^{(p-1)(q-1)} \equiv 1 \bmod p$ bzw. $p/(a^{(p-1)(q-1)} - 1)$,

$a^{(p-1)(q-1)} \equiv 1 \bmod q$ bzw. $p/(a^{(p-1)(q-1)} - 1)$.

Da p und q Primzahlen sind, folgt daraus

$pq/(a^{(p-1)(q-1)} - 1)$, also $a^{(p-1)(q-1)} \equiv 1 \bmod pq$ und daher

auch $a^{t(p-1)(q-1)} \equiv 1 \bmod pq$ für $t \in \mathbb{Z}$.

Daraus folgt die Behauptung $a^{1+t(p-1)(q-1)} \equiv a \bmod pq$.

Fall 2: Sei ggT (a,n) = p oder ggT(a,n) = q.
Ist o.B.d.A. ggT(a,n) = p, dann gilt wegen ggT(a,q) = 1 die Beziehung

$$a^{(p-1)(q-1)} \equiv 1 \bmod q,$$

also

$$a^{t(p-1)(q-1)} \equiv 1 \bmod q \quad \text{für } t \in \mathbb{Z}.$$

Aus $q/(a^{t(p-1)(q-1)} - 1)$ und p/a folgt aber nun

$pq/(a^{1+t(p-1)(q-1)} - a)$ bzw. $a^{1+t(p-1)(q-1)} \equiv a \bmod pq$ für $t \in \mathbb{Z}$.

Fall 3: Sei ggT(a,n) = n; dann ist die Behauptung wegen pq/a trivial.

Mit diesen Sätzen läßt sich nun die entsprechende Umkehrfunktion zu $c = m^k \bmod n$ leicht finden.
Wir müssen nur ein multiplikatives Inverses s zu k finden, sodaß $ks \equiv 1 \bmod (p-1)(q-1)$ gilt.

So ein s existiert genau dann, wenn $ggT(k,(p-1)(q-1)) = 1$.
Denn dann läßt sich, wie später gezeigt wird, s mit Hilfe des euklidischen Algorithmus berechnen.

Der Vollständigkeit halber sei noch gesagt, daß $(p-1)(q-1)$ dem Wert der Eulerschen Funktion $\phi(n)$ entspricht, wobei $n = p \cdot q$ ist.
($\phi(n)$ ist die Anzahl der zu n teilerfremden Zahlen von 1 bis n.)

Aus $ks \equiv 1 \bmod (p-1)(q-1)$ folgt $ks = 1+t(p-1)(q-1)$ für ein $t \in \mathbb{Z}$.

Damit gilt mit dem oben gezeigten Satz

$$(m^k)^s = m^{1+t(p-1)(q-1)} \equiv m \bmod n.$$

Wie man das zur Dechiffrierung notwendige s mit Hilfe des euklidischen Algorithmus berechnet, soll an unserem Beispiel gezeigt werden.

$k = 193, \quad p = 89, \quad q = 79, \quad (p-1)(q-1) = 88 \cdot 78 = 6864.$

Da $ggT(193,6864) = 1$, existieren Zahlen u,v aus $\mathbb{Z}$, sodaß $1 = u.193 + v.6864$ gilt.

Da aus $ks = 1 + t(p-1)(q-1)$ die Gleichung $1 = ks - t(p-1)(q-1)$ folgt, entspricht u genau unserem gesuchten s.
Die Zahlen u und v erhält man mit Hilfe des euklidischen Algorithmus.

Wir berechnen dazu $ggT(193,6864)$ mit Hilfe des euklidischen Algorithmus und stellen dabei gleichzeitig die entstehenden Reste als Linearkombinationen von 193 und 6864 dar:

$$ggT(193,6864) = ggT(109,193) = ggT(109,84) = ggT(25,84) = ggT(25,9) =$$

$$= ggT(7,9) = ggT(7,2)$$

$$
\begin{aligned}
109 &= -35\cdot 193 + 1\cdot 6864 \\
84 &= 36\cdot 193 - 1\cdot 6864 \\
25 &= -71\cdot 193 + 2\cdot 6864 \\
9 &= 249\cdot 193 - 7\cdot 6864 \\
7 &= -569\cdot 193 + 16\cdot 6864 \\
2 &= 818\cdot 193 - 23\cdot 6864
\end{aligned}
$$

Somit gilt $1 = -3023\cdot 193 + 85\cdot 6864$ und daher ist
$s = -3023 \bmod 6864 = 3841$.
Der Empfänger, der als einziger s kennt, dechiffriert durch $c_1^s \bmod 7031 = 4494^{3841} \bmod 7031 = 1001 = m_1$ und erhält so wieder die ursprüngliche Nachricht.

Wie wir gesehen haben läßt sich die Zahl s also berechnen, wenn p und q bzw. $\emptyset(n)$ bekannt sind. Wie man zeigen kann, würde jede andere Methode zur Berechnung so eines s auch zu einer Faktorisierung von n führen. Will ein Außenstehender die Nachricht lesen so müßte er zuerst n faktorisieren (oder ein dazu äquivalentes Problem lösen), was bei sehr großen p und q (etwa 100-stellig), aus Mangel an guten Algorithmen zur Faktorisierung, auch mit den besten Computern bis heute noch unmöglich ist. Theoretisch wäre dies natürlich möglich. Das RSA-Schema ist also kein absolut aber dennoch ein rechnerisch sicheres System.

Das RSA-Verfahren funktioniert also wie folgt:

Wähle zwei große Primzahlen p und q und bilde $p.q = n$.
Bestimme ein k mit $ggT(k,(p-1)(q-1)) = 1$.
Finde ein s mit $k,s \equiv 1 \bmod (p-1)(q-1)$. Nehme $M = C = \mathbb{Z}_n = \{0,1,2,\ldots n-1\}$.

RSA-Kryptosystem
Veröffentliche k und n.
Halte p,q und s geheim.
Verschlüsselung: $E(m) = m^k \bmod n = c$
Entschlüsselung: $D(c) = c^s \bmod n = m$

Signaturen mit dem RSA-Schema

Mit dem RSA-Schema sind sogar Signaturen möglich, da $(m^k)^s \bmod n = m^{ks} \bmod n = (m^s)^k \bmod n = m \bmod n$ gilt.
Es ist also egal, ob man mit k verschlüsselt und mit s entschlüsselt, oder umgekehrt. Schwierigkeiten können nur auftreten, wenn der Absender zuerst mit seinem Schlüssel s_A entschlüsselt und dann noch mit dem öffentlichen Schlüssel k_B des Empfängers verschlüsselt, und der Modul des Empfängers kleiner ist als der des Absenders.

Das folgende Beispiel soll die dabei auftretenden Probleme illustrieren:

A Absender:	$n_A = 7031$	$k_A = 193$	$s_A = 3841$
B Empfänger:	$n_B = 5183$	$k_B = 1483$	$s_B = 3667$

A möchte die signierte Nachricht "AS" an B senden.

$$m = \text{"AS"} = 0119$$

Er verschlüsselt m zuerst mit seinem privaten Schlüssel:

$$m^{s_A} \bmod n_A = 119^{3841} \bmod 7031 = 5460.$$

Die mit k_B zu verschlüsselnde Zahl ist also größer als der Modul n_B. Daher darf 5460 nicht mit k_B verschlüsselt werden, da für diesen Wert die Chiffrierfunktion nicht definiert ist.

A kann dieses Problem lösen, wenn er noch einmal mit s_A verschlüsselt, d.h. $5460^{3841} \bmod 7031 = 3344$ berechnet.

Man kann zeigen, daß man durch mehrfache derartige Verschlüsselungen stets eine Zahl erhält, die kleiner als der Modul des Empfängers ist.

In unserem Beispiel genügt das zweimalige Anwenden von S_A, und wir können jetzt mit k_B chiffrieren: $3344^{1483} \bmod 5183 = 4643$.

B entschlüsselt mit seinem privaten Schlüssel s_B und anschließend mit dem öffentlichen Schlüssel k_A und erhält 5460. Diese Nachricht ist aber größer als n_B, also muß er nochmals mit k_A entschlüsseln und erhält die Nachricht m = 119 = "AS".

Eine andere Möglichkeit wäre, daß man im Fall $m_A > m_B$ signiert, indem man $c = s_A(k_B(m))$ bildet. Also einfach, indem man zuerst mit k_B chiffriert und dann erst s_A anwendet.

Das RSA-Schema ist ein sehr brauchbares System, das vielen Anforderungen standhält und bereits in vielen Bereichen verwendet wird. Da es jedoch ein relativ langsames Verfahren ist, wird es vorwiegend zum Austausch kurzer Nachrichten verwendet, so vor allem auch zur Mitteilung von Schlüsseln für klassische Chiffrierverfahren.

Wir bringen nun noch ein weiteres Beispiel für ein RSA-System, an dem wir einige Probleme bei der Schlüsselwahl aufzeigen wollen, ohne diese hier ausführlich diskutieren zu können:

$p = 5$, $q = 11$, $n = 55$
$k = 3$, g.g.T. $(3,40) = 1$, $s = 27$ $(3 \cdot 37 = 81 \equiv 1 \bmod 40)$
$M = \{0,1,2,\ldots,54\}$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$x^3 \bmod 55$	0	1	8	27	9	15	51	13	17	14	10	11	23	52

x	14	15	16	17	18	19	20	21	22	23	24	25	26	27
$x^3 \bmod 55$	49	20	26	18	2	39	25	21	33	12	19	5	31	48

x	28	29	30	31	32	33	34	35	36	37	38	39	40	41
$x^3 \bmod 55$	7	24	50	36	43	22	34	30	16	53	37	29	35	6

x	42	43	44	45	46	47	48	49	50	51	52	53	54	
$x^3 \bmod 55$	3	32	44	45	41	38	42	4	40	46	28	47	54	

Tabelle 5

Die Abbildung $x \to x^3 \bmod 55$ besitzt 9 Fixpunkte. Fixpunkte sind aber sehr störend.

Zur Vermeidung von zu vielen Fixpunkten (9 ist die Minimalanzahl), sind kompliziertere Anweisungen für p und q zu beachten. "Gute Schlüssel" für ein RSA-Verfahren müssen heute die Bedingungen p und q prim, $\frac{p-1}{2}$ und $\frac{q-1}{2}$ prim, p+1 und q+1 besitzen große Primteiler, erfüllen.

Nach dem Satz von Euler-Fermat gilt $x^{81} \equiv x \bmod 55$ für alle $x \in \mathbb{Z}_{55}$. Rechnet man nach, sieht man jedoch, daß auch schon $x^{21} \equiv x \bmod 55$ für alle $x \in \mathbb{Z}_{55}$. Man kann daher auch schon $s = 7$ wählen, was die Rechnung wesentlich vereinfacht.

Man kann zeigen, daß es genügt s so zu wählen, daß $ks \equiv 1 \bmod kgV\,(p-1, q-1)$ ist.

Tabelle 6 gibt einen Überblick über die Fortschritte beim Faktorisieren großer Zahlen.

Faktorisieren großer Zahlen		
Jahr	Anzahl der Dezimalstellen	Rechenzeit
1970	42	
1980	49	
1982	55	
1984	71	7 Stunden
	75	1 Tag
	100	ca 10^5 Jahre

Tabelle 6

Zur Erzeugung der Schlüssel für ein RSA-Verfahren mit 100-stelligen Primzahlen benötigte man 1980 etwa 20 Minuten. Heute kann man sehr gute Schlüssel (man wählt sehr spezielle Pseudoprimzahlen um mögliche Attacken zu vermeiden) in bereits 5 Sekunden generieren. Eine erste RSA-Implementierung (Sandia I) rechnete 420 Bits pro Sekunde. Heute verarbeitet man 2 000 Bits pro Sekunde (RSA-Security Inc.).

5. MH-KRYPTOSYSTEM (Merkle/Hellman (1978))

Zum Abschluß wollen wir noch das sogenannte MH-Schema kurz diskutieren, obwohl dieses Kryptosystem 1982 von Shamir (vgl. Shamir (1983)) mit einem Polynomzeitalgorithmus gebrochen werden konnte. Die "Schwäche" dieses Systems ist aber nicht das Problem, auf dem es basiert, sondern es sind die dabei verwendeten modularen Transformationen und die speziellen Annahmen, die man trifft. Das zugrundeliegende "Untersummenproblem" (Knapsack-Problem) muß sogar als eines der härtesten Probleme der Mathematik angesehen werden und es wird heute vielfach zur Konstruktion von Falltürfunktionen verwendet. So liegt Chiffrierungen für Funkverbindungen der Europäischen Raumfahrtsbehörde angeblich das Knapsack-Problem zugrunde.

In den letzten Jahrzehnten hat sich in der Mathematik eine Reihe von Problemen herausgeschält, zu deren Lösung man trotz jahrelanger Bemühungen keinen "guten" Algorithmus gefunden hat, d.h. keinen Algorithmus, welcher nicht exponentiell mit dem Umfang der Aufgabe anwächst. Fast alle diese Probleme sind kombinatorische Optimierungsprobleme und man konnte vor wenigen Jahren zeigen, daß sie rechnerisch äquivalent sind. Wenn es also für eines dieser Probleme einen guten Algorithmus gibt, dann gibt es für alle einen guten Algorithmus. Man nennt diese Probleme NP-vollständig. Heute ist man davon überzeugt, daß es für NP-vollständige Probleme keinen guten Algorithmus gibt.

M. Hellman hatte die Idee, NP-vollständige Probleme zur Konstruktion öffentlicher Chiffriersysteme zu verwenden. Man konstruiert dabei die Funktionen E_X und D_X so, daß die Bestimmung von D_X aus E_X die Lösung eines NP-vollständigen Problemes erfordert.

Dem folgenden MH-Verfahren liegt das NP-Problem zugrunde, aus einer vorgegebenen Menge von Zahlen eine Teilmenge mit gegebener Summe zu finden. Zur Lösung dieser Aufgabe muß man im wesentlichen alle 2^n Teilmengen der gegebenen n-elementigen Menge durchprobieren. Wie beim RSA-Verfahren werden auch beim

MH-Verfahren die Nachrichten zunächst einmal mit Hilfe eines binären Alphabets (vgl. Tabelle 3) geschrieben und dann in Blöcke der Länge n zerlegt:

$$m = x_1x_2x_3x_4x_5 \; x_6x_7x_8x_9x_{10}\cdots = x_1x_2\cdots x_n \; x_{n+1}x_{n+2}\cdots$$

Jede Person, welche geheime Nachrichten empfangen möchte, wählt dann einen Vektor $(a_1', a_2', \ldots, a_n')$ mit Komponenten a_i' aus den natürlichen Zahlen $\mathbb{N}$ $(i=1,2,\ldots,n)$ und

$$a_i' > (a_1' + a_2' + \ldots + a_{i-1}') \text{ für } i = 2,3,\ldots,n.$$

Weiters wird eine natürliche Zahl $t > a_1' + \ldots + a_n'$ gewählt und ein natürliches k mit $ggT(k,t) = 1$ bestimmt. Nun berechnet man die Zahlen a_i aus $a_i \equiv ka_i' \bmod t$, $0 \; a_i < t$.

Zum Verschlüsseln eines Nachrichtenblockes $(x_1,\ldots,x_n)$ bildet man nun die Summe

$$\sum_{i=1}^{n} a_i x_i := \bar{a}.$$

Um $\bar{a}$ zu entschlüsseln benötigt man eine natürliche Zahl s mit $sk \equiv 1 \bmod t$. Dann gilt nämlich

$$s\bar{a} = \sum_{i=1}^{n} sa_i x_i \equiv \sum_{i=1}^{n} a_i' x_i \bmod t$$

und da

$$\sum_{i=1}^{n} a_i' x_i \leq \sum_{i=1}^{n} a_i' < t,$$

sehen wir, daß $\sum_{i=1}^{n} a_i' x_i$ der kleinste nichtnegative Rest bei Division von $s\bar{a}$ durch t ist. Nach der Konstruktion der a_i' ist x_n der Quotient und $\sum_{i=1}^{n-1} a_i' x_i$ der Rest bei Division von $\sum_{i=1}^{n} a_i' x_i$ durch a_n', und x_{n-1} ist der Quotient und $\sum_{i=1}^{n-2} a_i' x_i$ der Rest bei Division von $\sum_{i=1}^{n-1} a_i' x_i$ durch a_{n-1}', usw.

Mit Kenntnis von t und s kann man also aus $(a_1,\ldots,a_n)$ den Vektor $(a_1',\ldots,a_n')$ berechnen und damit aus der empfangenen verschlüsselten Zahl $\bar{a}$ die ursprüngliche Nachricht $(x_1,\ldots,x_n)$. Ohne Kenntnis von t und s muß man das Problem lösen, eine Teilsumme aus den Komponenten von $(a_1',\ldots,a_n')$ zu finden welche $\bar{a}$ ergibt. Bildet man den "Knapsack-Vektor" aus 200 20-stelligen Zahlen und wählt k,s und t 30 bis 60-stellig, dann glaubte man bis Shamir (1983) dieses Problem auch auf den schnellsten und größten derzeit zur Verfügung stehenden Rechenmaschinen nicht bewältigen zu können. Ein großer Vorteil des MH-Kryptosystems ist seine Schnelligkeit. Beim MH-Verfahren besteht der öffentliche Schlüssel E in der Angabe der Kodierungsart und dem Vektor $(a_1,\ldots,a_n)$. Der private Schlüssel D besteht aus t und s.

Wir wollen auch hier wieder ein kleines Zahlenbeispiel angeben: Sei $(a_1',\ldots,a_6') = (3,5,11,20,41,83)$, $t = 170$ und $k = 73$. Dann bekommt man aus

$$a_i = 73\, a_i' \bmod 170, \quad i = 1,2,\ldots,6,$$

den Vektor $(a_1,\ldots,a_6) = (49,25,123,100,103,109)$. Dieser Vektor wird als öffentlicher Schlüssel publiziert. Möchte nun jemand die Nachricht m = "JA" = 01010 00001 senden, so verschlüsselt er

$$49\cdot 0 + 25\cdot 1 + 123\cdot 0 + 100\cdot 1 + 103\cdot 0 + 109\cdot 0 = 125$$

und sendet diese Zahl. Der Empfänger berechnet sich ein geeignetes s, z.B. $s = 7$, und bildet $7\cdot 125 = 875 = 25 \bmod 170$. Damit läßt sich sofort $(x_1,x_2,\ldots,x_6) = (0,1,0,1,0,0)$ bestimmen. Für $x_7,x_8,\ldots$ wird analog vorgegangen. (Eventuell freibleibende Stellen bei der Einteilung der Nachricht in Blöcke der Länge 6 kann man durch Nullen auffüllen.)

Wie schon erwähnt, gibt es einen Polynomzeitalgorithmus, welcher das MH-Kryptosystem entschlüsselt. Die Schwäche des Verfahrens liegt in der speziellen Wahl des Vektors $(a_1',\ldots,a_n')$. Das allgemeine Knapsack-Problem ist nach wie vor ungelöst.

ZITIERTE LITERATUR

Diffie, W. und Hellman, M.E.: New Directions in Cryptography. IEEE Transactions on Information Theory, vol. IT-22, No. 6, 644-654, 1976.

Eier, R. und Lagger, H.: Trapdoors in Knapsack-Cryptosystems. Lecture Notes in Computer Science, vol. 149, 316-322, 1983.

Ingemarsson, I.: A New Algorithm for the Solution of the Knapsack Problem. Lecture Notes in Computer Science, vol. 149, 309-315, 1983.

Katzan, H.(Jr.): The Standard Data Encryption Algorithm. Petrocelli Books, New York, 1977.

Massey, J.L. und Omura, J.K.: A New Multiplication Method for $GF(2^m)$ and its Application in Public-Key Cryptography. Vortragsmanuskript Eurocrypt '83, Udine, Italien, 1983.

Mc. Eliece, R.J.: A Public-Key Cryptosystem Based on Algebraic Coding Theory. DNS Progess Report 42-44, 114-116, 1978.

Merkle, R.C. und Hellman, M.E.: Hiding Information and Signatures in Trapdoor Knapsacks. IEEE Transactions on Information Theory 24, 525-530, 1978.

Müller, W.B. und Nöbauer, W.: Some Remarks on Public-Key Cryptosystems. Studia Sci.Math.Hungar. 16, 71-76, 1981.

Nöbauer, R.: Rédei-Funktionen und ihre Anwendung in der Kryptographie. Erscheint in Acta Sci. Math. Szeged, 1986.

Rabin, M.O.: Probabilistic Algorithm for Primality Testing. J. Number Theory 12, 128-138, 1980.

Rivest, R.L., Shamir, A. und Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, vol. 21, 120-126, 1978.

Shamir, A.: A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem. Advances in Cryptology, Plenum Press, New York, 1983.

Sloane, N.J.A.: Error Correcting Codes and Cryptography. The Mathematical Gardner, ed. by D.A.Klarner; Prindle, Webster and Schmidt, Boston, 346-382, 1981.

Empfehlenswerte Literatur zur Kryptographie

Beker, H. und Piper, F.: Cipher Systems. Northwood Books, London, 1982.

Beth, Th., Heß, P. und Wirl, K.: Kryptographie. B.G.Teubner, Stuttgart, 1983.

Dorninger, D. und Müller, W.B.: Allgemeine Algebra und Anwendungen. B.G.Teubner, Stuttgart, 1984.

Engel, A.: Datenschutz durch Chiffrieren: Mathematische und algorithmische Aspekte. MU 6, 30-51, 1979.

Franke, H.W.: Die geheime Nachricht. Umschau Verlag, Frankfurt, 1982.

Hellman, M.E.: Die Mathematik neuer Verschlüsselungssysteme. Spektrum der Wissenschaft 10, 92-101, 1979.

Kahn, D.: The Codebreakers. Macmillan, New York und London, 1967.

Kohnheim, A.G.: Cryptography, A Primer. Wiley-Interscience, New York, 1982.