Einführung in die Algebra der Public-Key-Kryptographie

Timm Grams, Fulda, 24. Januar 2012 (rev. 26.01.2012)

Inhalt

Einführung ... 1
Zweck ... 1
Literaturhinweise ... 2
Modulo-Rechnung ... 2
Ganzzahlige Division ... 2
Rechnen mit Resten ... 2
Rechnen mit Restklassen ... 4
Grundlegende Beziehungen der Algebra ... 4
Erweiterter euklidischer Algorithmus ... 4
Auflösen von Gleichungen der Modulo-Rechnung ... 5
Einige endliche Körper und ein Puzzle ... 5
Das Merkle-Hellman-Kryptosystem ... 7
Public-Key-Kryptographie ... 7
Konstruktion von privatem und öffentlichem Schlüssel ... 8
Codierung ... 8
Problem des Lauschers ... 8
Decodierung ... 9
Kritik ... 9
Chinesischer Restsatz ... 9
Der Satz ... 9
Konstruktiver Existenzbeweis ... 10
Beispiel ... 10
Beweis der Eindeutigkeit der Lösung ... 11
Die eulersche Funktion φ ... 11
Die Sätze von Pierre de Fermat und Leonhard Euler ... 12
Die Gruppe der zum Modul teilerfremden Reste ... 12
Untergruppen und Restklassen ... 13
Das RSA-Kryptosystem ... 14

Einführung

Zweck

Populärwissenschaftliche Darstellungen der in der Internetkommunikation weit verbreiteten Verschlüsselung mit öffentlichem Schlüssel (Public Key Cryptography) müssen sich hinsichtlich der algebraischen Grundlagen knapp fassen. Andererseits ist ein gründliches Studium der algebraischen Grundlagen recht aufwendig und im Rahmen eines Informatik- oder Technikstudiums kaum durchführbar. Hier wird versucht, diese Lücke zu schließen und die Algebra der Verschlüsselungssysteme in einer für Ingenieur- und Informatikstudenten angemessenen Weise darzustellen. Dabei werden einige der mathematischen Sätze nur für Fälle bewiesen, die für das Verständnis der Public-Key-Kryptographie wichtig sind. Der allgemeine Fall bleibt aber im Blickfeld, so dass der Beweis vom Leser leicht verallgemeinert werden kann. Bei der Definition mathematischer Opterationen lehne ich mich an die Begriffsbildung heutiger Programmiersprachen an (Pascal, C, Java). Der Text kann auch als Vorbereitungskurs für ein Studium der Algebra für Ingenieure und Informatiker dienen.

Literaturhinweise

Aho, Alfred V.; Hopcroft, John E.; Ullman, Jeffrey D.: The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, Massachusetts 1974 *(Stellt den Zusammenhang zwischen dem chinesischen Restsatz und der schnellen Fourier-Transformation her.)*

Grams, T.: Codierungsverfahren. Bibliographisches Institut, Mannheim 1986. *(Beispiele für das Rechnen im binären Körper.)*

Hellman, Martin E.: Die Mathematik neuer Verschlüsselungssysteme. Spektrum der Wissenschaft (1979) 10, 92-101 *(Populärwissenschaftliche Darstellung der hier behandelten Public-Key-Kryptographieverfahren.)*

Körner, Otto: Algebra. Akademische Verlagsgesellschaft, Frankfurt am Main 1074. *(An Körners Darstellung der algebraischen Grundlagen lehnt sich meine Darstellung an.)*

Rivest, R. L.; Schamir, A.;Adleman, L.: A method for Obtaining Digital Signatures and Public-Key Cryptosystems. Comm. of the ACM 21(1978) 2, 120-126. *(Originalarbeit zum RSA-Verfahren.)*

Van der Waerden, B. L.: Algebra I. Springer-Verlag, Berlin, Heidelberg, New York 1971. *(Standardwerk zur Algebra.)*

Wikipedia-Themen (deutsch): "Merkle-Hellman-Kryptosystem", „RSA-Kryptosystem“

Wikipedia-Themen (englisch): "Merkle-Hellman knapsack cryptosystem", „RSA (Algorithm)“

Modulo-Rechnung

Ganzzahlige Division

Basis der modernen Verschlüsselungssysteme bildet die Modulo-Rechnung. Nehmen wir zwei ganze Zahlen a und b mit $b\neq 0$. Wir stellen die Zahl a als Vielfaches der Zahl b zuzüglich eines nichtnegativen Rests dar, der kleiner als b ist:

$$a = b\cdot q + r \text{ mit } 0\leq r<|b|. \tag{0}$$

Beispiele: $23 = 5\cdot 4 + 3$, $-23 = 5\cdot(-5) + 2$, $-44= (-2)\cdot 22 + 0$.

Der ganzzahlige Quotient q und der Rest r sind eindeutig bestimmt und wir schreiben

$$q = a \text{ div } b$$

und

$$r = a \bmod b$$

(gesprochen: a modulo b). Es gilt die *Grundgleichung der ganzzahligen Arithmetik*

$$a = b\cdot(a \text{ div } b) + a \bmod b. \tag{0'}$$

Zumindest für nichtnegative Werte a und b hat a div b die Bedeutung der ganzzahligen Division mit dem Rest a mod b. Einige Schwierigkeiten macht die Interpretation dieser Gleichung im Bereich negativer Zahlen[1]. Aber das ist hier ohne Belang: Wir definieren für den gesamten Zahlenbereich den div-Operator durch die obige Grundgleichung, auch wenn es für negative Zahlen etwas quietscht.

Rechnen mit Resten

Wir halten nun den Divisor fest und bezeichnen ihn mit m. Für das Rechnen mit Resten modulo m gelten diese Regeln:

$$((a \bmod m) + (b \bmod m)) \bmod m = (a + b) \bmod m, \tag{1}$$

$$((a \bmod m)\cdot(b \bmod m)) \bmod m = a\cdot b \bmod m. \tag{2}$$

[1] In den Sprachen C und Java behält der div-Operator auch im Bereich negativer Zahlen die Bedeutung der ganzzahligen Division. Der Rest kann dann negativ werden. Um die Grundgleichung der ganzzahligen Arithmethik zu retten, muss dafür die mod-Operation für den Bereich negativer Zahlen anders definiert werden als hier. In der Sprache Pascal nach Nikolaus Wirth bleibt es hingegen bei der oben definierten mod-Operation. Gute Gründe sprechen dafür. Siehe dazu meinen Beitrag zum Artikel von Jürg Nievergelt: "Über das div-mod-Problem und die Normierung ganzzahliger Arithmetik sowie ein Rückblick auf Zahlenkreuze". Informatik-Spektrum (1991) 14, S. 351-354.

Bevor es an den Beweis dieser Gleichungen geht, will ich noch eine Kurzschreibweise einführen. Solange klar ist, mit welchem Modul m die Operatoren gebildet werden, schreiben wir $\bar{a}$ für a mod m und $\underline{a}$ für a div m. Die Grundgleichung wird so zu $a = m\cdot\underline{a} + \bar{a}$ und die zu beweisenden Gleichungen werden zu

$$\overline{\bar{a}+\bar{b}} = \overline{a+b}, \tag{1′}$$

$$\overline{\bar{a}\cdot\bar{b}} = \overline{a\cdot b}. \tag{2′}$$

Beweis: Wir benutzen die Grundgleichungen.

$$a = m\cdot\underline{a} + \bar{a},$$

$$b = m\cdot\underline{b} + \bar{b}.$$

Im Fall der Addition haben wir

$$a+b = m\cdot\underline{a+b} + \overline{a+b}.$$

Daraus folgt

$$\begin{aligned}
\overline{a+b} &= a+b-m\cdot\underline{a+b} && \text{// vorhergehenden Zeile umgestellt}\\
&= m\cdot\underline{a} + \bar{a} + m\cdot\underline{b} + \bar{b} - m\cdot\underline{a+b} && \text{//}a, b \text{ ersetzt}\\
&= \bar{a} + \bar{b} + m\cdot(\underline{a} + \underline{b} - \underline{a+b}). && \text{//Umordnung}
\end{aligned}$$

Wegen $\bar{a} + \bar{b} = m\cdot\underline{\bar{a}+\bar{b}} + \overline{\bar{a}+\bar{b}}$ folgt daraus die Gleichung

$$\overline{a+b} = \overline{\bar{a}+\bar{b}} + m\cdot(\underline{\bar{a}+\bar{b}} + \underline{a} + \underline{b} - \underline{a+b}).$$

Die beiden Modulo-Ausdrücke sind nichtnegativ und kleiner als $|m|$. Der Betrag ihrer Differenz ist folglich ebenfalls kleiner als $|m|$. Das verlangt, dass der geklammerte Ausdruck gleich null ist. Damit sind (1) und (1′) bewiesen.

Im Fall der Multiplikation haben wir

$$ab = m\cdot\underline{ab} + \overline{ab}.$$

Daraus folgt

$$\begin{aligned}
\overline{ab} &= ab-m\,\underline{ab} && \text{// vorhergehenden Zeile umgestellt}\\
&= (m\,\underline{a} + \bar{a})(m\,\underline{b} + \bar{b}) - m\,\underline{ab} && \text{//}a, b \text{ ersetzt}\\
&= \bar{a}\,\bar{b} + m(\underline{a}\,\bar{b} + \bar{a}\,\underline{b} + m\,\underline{a}\,\underline{b} - \underline{ab}). && \text{//Umordnung}
\end{aligned}$$

Wegen $\bar{a}\,\bar{b} = m\cdot\underline{\bar{a}\bar{b}} + \overline{\bar{a}\bar{b}}$ ergibt sich daraus die Gleichung

$$\overline{ab} = \overline{\bar{a}\bar{b}} + m(\underline{\bar{a}\bar{b}} + \underline{a}\,\bar{b} + \bar{a}\,\underline{b} + m\,\underline{a}\,\underline{b} - \underline{ab}).$$

Mit derselben Begründung wie oben erhalten wir (2) und (2′).

Die Ausdrücke $\overline{\bar{a}+\bar{b}}$ und $\overline{\bar{a}\cdot\bar{b}}$ definieren die Addition und Multiplikation für Reste. Wenn keine Missverständnisse zu befürchten sind, werden für a und b die Reste selbst eingesetzt und die Überstreichungen weggelassen. Das Rechnen mit den Resten zu einem festen Modul m genügt denselben Rechenregeln wie das Rechnen mit ganzen Zahlen. Das lässt sich leicht

mit Hilfe der Formeln (1′) und (2′) zeigen. Insbesondere die Regeln der Klammernrechnung bleiben gültig.

Rechnen mit Restklassen

Wir interpretieren die Modulo-Bildung nun etwas um. Wir fassen alle Zahlen, die bei Division denselben Rest haben, zu einer Menge zusammen und bezeichnen diese Klasse genauso wie den Rest selbst. Wir setzen also $\bar{a} = \{b \mid \bar{b} = \bar{a}\}$. In der geschweiften Klammer steht die Modulo-Bildung, wie wir sie oben eingeführt haben. Vor dem Gleichheitszeichen bezeichnet dasselbe Symbol die Restklasse von *a* modulo *m.*

Jedes Element einer Restklasse kann offenbar die gesamte Menge repräsentieren. Für je zwei Elemente *a* und *b* einer Restklasse modulo *m* gilt offenbar *a*-*b* = *mk* mit einer ganzen Zahl *k*. Und immer wenn für zwei Zahlen dieser Zusammenhang gilt, gehören sie derselben Restklasse modulo *m* an und es gilt $\bar{a} = \bar{b}$ auch für die jeweils zugehörigen Restklassen.

Die Formeln (1), (1′), (2) und (2′) definieren damit auch die Addition und die Multiplikation für Restklassen. Ob wir nun mit Restklassen rechnen oder mit den Resten: Es ist nur eine Sache der Interpretation der Elemente, nichts weiter. Mengen mit additiver und multiplikativer Verknüpfung, die den Rechenregeln für ganze Zahlen genügen, bilden einen *kommutativen Ring mit Einselement.*

Unter den ganzen Zahlen haben nur die Einheiten 1 und -1 ein Inverses bezüglich der Multiplikation. Diese Einschränkung gilt für Restklassen nicht unbedingt.

Interessant sind die Fälle, in denen die Restklassen-Gleichung *AX*= 1 bei gegebenem *A* eine Lösung *X* besitzt. In dieser Gleichung haben wir die Restklasse von 1 modulo *m*, also $\bar{1}$, wieder mit 1 bezeichnet. In welchen Fällen sich die Gleichung lösen lässt, werden wir mit dem erweiterten euklidischen Algorithmus klären.

Die Lektüre und das Verstehen der folgenden Seiten fallen leichter, wenn man sich vor Augen hält, dass die folgenden Gleichungen bzw. Aussagen alle dasselbe besagen.

$\bar{x} = \bar{y}$

$x \bmod m = y \bmod m$

$(x\text{-}y) \bmod m = 0$

m ist Teiler von *x*–*y*

Es gibt ein *k,* sodass $x\text{-}y = k \cdot m$ gilt.

Grundlegende Beziehungen der Algebra

Erweiterter euklidischer Algorithmus

Gegeben seien zwei positive Zahlen *a* und *b*. Für das Beispiel wähle ich 666 und 246. Diese beiden Zahlen werden in die entsprechend benannten Spalten 2 und 3 einer Tabelle eingetragen und umbenannt in a_1 und b_1. Die Spalte 1 dient der Nummerierung der Tabellenzeilen. Die Spalte 3 enthält den Quotienten *q* der ganzzahligen Division von *a* und *b.* Für sämtliche Zeilen *i* gilt $q_i = a_i \text{ div } b_i$. Die Spalten für *a, b* und *q* sind durch folgende Festlegung vollständig rekursiv definiert: $a_{i+1} = b_i$ und $b_{i+1} = a_i \bmod b_i$. Aus den Startwerten a_1 und b_1 ergeben sich in der *Vorwärtsrechnung* die übrigen Werte über diese Rekursionsgleichungen:

$$a_i = b_i \cdot q_i + b_{i+1},$$
$$a_{i+1} = b_i.$$

i	*a*	*b*	*q*	*s*	*t*
1	666	246	2	17	-46
2	246	174	1	-12	17
3	174	72	2	5	-12
4	72	30	2	-2	5
5	30	12	2	1	-2
6	12	6	2	0	1
7	6	0		1	0

Dabei sind – bei gegebenen a_i und b_i – die Werte q_i und b_{i+1} durch die Bedingung $0 \le b_{i+1} < b_i$ eindeutig festgelegt, siehe Gleichung (0). Die Rekursion bricht ab, sobald eine Division durch null droht. Das ist in der letzten Zeile der Tabelle der Fall.

Damit haben wir eine Tabellenform des euklidischen Algorithmus vor uns. Jeder gemeinsame Teiler von a und b ist auch Teiler sämtlicher a_i und b_i, wie man den Rekursionsgleichungen entnehmen kann. Die Zahl in der letzten Zeile der a-Spalte, hier $a_6 = 6$, enthält diesen Teiler ebenfalls. Offenbar teilt diese Zahl sämtliche a_i und b_i. Folglich muss diese Zahl der größte gemeinsame Teiler von a und b sein, kurz ggT(a, b).

Die Werte der s- und die t-Spalte werden so bestimmt, dass für jede Zeile der Tabelle die *Invariante*

$$\text{ggT}(a,b) = a_i \cdot s_i + b_i \cdot t_i$$

gilt. Diese Bedingung und die Gleichungen für die Vorwärtsrechnung liefern die Gleichungen für die *Rückwärtsrechnung:*

$$s_i = t_{i+1},$$
$$t_i = s_{i+1} - q_i \cdot t_{i+1}.$$

Damit hangelt man sich von der letzten bis zur ersten Zeile der Tabelle vor und erhält schließlich für die erste Zeile mithilfe der Invariantengleichung und unter Verwendung der Abkürzungen $s = s_1$ und $t = t_1$ das wichtige Ergebnis

$$\text{ggT}(a, b) = a \cdot s + b \cdot t.$$

Beispiel: $6 = 666 \cdot 17 + 246 \cdot (-46)$.

Also: Der größte gemeinsame Teiler zweier Zahlen lässt sich als ganzzahlige Kombination dieser Zahlen darstellen. Interessant ist der Fall, dass die Zahlen teilerfremd sind. Dann gilt nämlich ggT(a, b) = 1 und

$$a \cdot s + b \cdot t = 1.$$

Auflösen von Gleichungen der Modulo-Rechnung

Nun können wir leicht einsehen, wann die Restklassengleichung $AX = 1$ auflösbar ist. Sei a ein Repräsentant der Restklasse A: $A = \bar{a}$. Seien ferner a und der Modul m teilerfremd. Dann liefert der erweiterte euklidische Algorithmus zwei Werte s und t derart, dass

$$a \cdot s + m \cdot t = 1.$$

Das heißt, dass $a \cdot s$ und 1 derselben Restklasse modulo m angehören. Wir setzen $X = \bar{s}$ und erhalten $AX = \overline{a \cdot s} = \bar{1} = 1$. Damit haben wir das multiplikativ inverse Element zu A gefunden:

$$A^{-1} = X = \bar{s}.$$

Einige endliche Körper und ein Puzzle

Die Reste bezügliche eines bestimmten Moduls – oder auch die Restklassen – bilden mit den Operationen der Addition und Multiplikation einen kommutativen Ring mit Einselement. Bei geeigneter Wahl des Moduls m gelten sogar die Regeln für das Auflösen von Gleichungen: Zu jedem Element ungleich 0 gibt es ein inverses Element und folglich ist die Gleichung $AX = B$

für alle $A \neq 0$ nach X auflösbar. Bei solchen Resteringen oder Restklassenringen handelt es sich demnach sogar um *kommutative Körper*.

Jedenfalls ergeben sich immer dann Körper, wenn der Modul m eine Primzahl ist. Die Verknüpfungstabellen für den Körper der Binärzeichen sehen so aus ($m = 2$):

+	**0**	**1**
0	0	1
1	1	0

·	**0**	**1**
0	0	0
1	0	1

Die Multiplikation entspricht der logischen AND- und die Addition der XOR-Verknüpfung. Jedes Element ist zu sich selbst additiv invers. Für die Multiplikation gilt dasselbe für das einzige Element ungleich 0: $1^{-1} = 1$. Der *binäre Körper* spielt für die redundante Codierung eine zentrale Rolle (Grams, 1986).

Für den ternären Körper erhalten wir diese Verknüpfungstabellen:

+	**0**	**1**	**2**
0	0	1	2
1	1	2	0
2	2	0	1

·	**0**	**1**	**2**
0	0	0	0
1	0	1	2
2	0	2	1

Die Inversen Elemente sind $-0 = 0$, $-1 = 2$, $-2 = 1$, $1^{-1} = 1$, $2^{-1} = 2$.

Wegen der Körpergesetze lassen sich unter anderem Gleichungen genauso auflösen, wie im Bereich der reellen Zahlen: Wir können die Matrizenrechnung auf die neuen Zahlen übertragen. Der *gaußsche Algorithmus* zur Auflösung von Gleichungen steht zur Verfügung und nichtsinguläre Matrizen besitzen Inverse. Diese Tatsache lässt sich vorteilhaft bei der Lösung des folgenden Puzzles („Es werde Licht“) nutzen. (Es gibt elementare Lösungen, aber diese verlangen einen höheren Argumentationsaufwand.)

Aufgabenstellung: Sieben Lampen sind kreisförmig angeordnet. Zu jeder Lampe gehört ein Taster zum Ein- und Ausschalten, wie bei einer Nachttischlampe. Mit einem Tastendruck wird die zugeordnete Lampe umgeschaltet. Dummerweise aber nicht nur die, sondern mit ihr die beiden benachbarten Lampen. Gegeben ist ein Startzustand: Einige der Lampen sind an, einige aus. Die Aufgabe lautet, mit möglichst wenig Tastendrücken alle Lampen anzuschalten. Gesucht ist eine Strategie, die bei jedem beliebigen Anfangszustand den erwünschten Erfolg hat.

Lösungsvorschlag: Die Lampen werden von 1 bis 7 durchnummeriert. Der Zustand der Lampen wird durch einen Vektor y dargestellt: $y = (y_1, y_2, y_3, y_4, y_5, y_6, y_7)$. Die 1 steht für den Aus-Zustand und die 0 für den Ein-Zustand. Allgemeiner: Der Vektor markiert genau die Lampen mit einer 1, die umgeschaltet werden sollen; im Fall $y = 1000010$ sind genau die erste und die sechste Lampe zu schalten[2]. Analog werden die für die Zustandsänderung nötigen Schaltkombinationen durch den Vektor x repräsentiert: $x = (x_1,x_2,x_3,x_4, x_5, x_6, x_7)$; $x = 0110001$ heißt, dass die Schalter 2, 3 und 7 betätigt werden.

[2] Ohne Missverständnisse befürchten zu müssen, werden in den Vektoren zwischen den Ziffern 0 und 1 die Trennzeichen weggelassen, ebenso die begrenzenden Klammern.

Die Beziehung zwischen x und y werden durch eine Matrix A hergestellt: $y^T = A \cdot x^T$. Diese Matrix drückt aus, dass eine Lampe durch den ihr direkt zugeordneten Schalter als auch von den direkt benachbarten geschaltet werden kann. Wenn eine ungerade Zahl dieser Schalter betätigt wird, wechselt der Lampenzustand, sonst nicht. Diese Matrix besitzt eine Inverse, ich nenne sie B. Diese Matrix lässt sich mit dem Eliminationsverfahren von Gauß ermitteln. Die Lösung unseres Problems ist damit gegeben durch $x^T = B \cdot y^T$.

Die Matrix *A:*

```
1100001
1110000
0111000
0011100
0001110
0000111
1000011
```

Jedem Schalter ordne ich nun eine *Lampenauswahl* zu. Sie wird durch das entsprechende Zeilenmuster der Matrix B bestimmt. Zur Lampenauswahl eines Schalters gehören die ihm direkt zugeordnete Lampe und die beiden benachbarten Lampen. Außerdem werden die dem Schalter gegenüber liegenden beiden Lampen hinzugerechnet. Zur Lampenauswahl des Schalters 1 gehören also die Lampen 1, 2, 4, 5 und 7. Die Regel für die Lampenauswahl ist eine *Invariante* der Zeilen der Matrix A.

Die Matrix *B:*

```
1101101
1110110
0111011
1011101
1101110
0110111
1011011
```

Regel: Ist die Zahl der zu schaltenden Lampen in einer Lampenauswahl ungerade, dann muss der betreffende Schalter betätigt werden, sonst nicht.

Das Merkle-Hellman-Kryptosystem

Public-Key-Kryptographie

Nachrichten sollen vor Lauschern geschützt übertragen werden. In öffentlichen Übertragungsnetzen ist immer damit zu rechnen, dass der Lauscher die übermittelte Nachricht mithören kann. Um den Lauschangriff abzuwehren, benutzt der Sender ein Verschlüsselungsverfahren E (Encoding). Damit wandelt er seine ursprüngliche Nachricht x in eine Nachricht $y = E(x)$ um. Die so codierte Nachricht y übermittelt er über das öffentliche Netz an den Empfänger. Der Empfänger kennt das Entschlüsselungsverfahren D (Decoding). Diese Funktion ist die Inverse zu E. Folglich kann er die ursprüngliche Nachricht aus der verschlüsselt übertragenen Nachricht zurückgewinnen:

$$D(y) = D(E(x)) = x.$$

Der Lauscher kennt den Dekodierungsschlüssel D nicht, vorausgesetzt, es wird ein Verfahren verwendet, das sich nicht so ohne Weiteres aus der verschlüsselten Nachricht rekonstruieren lässt.

Problematisch bei dieser Sache ist, dass es zwischen Sender und Empfänger zu einer Vereinbarung über das Verschlüsselungsverfahren kommen muss. Und auch diese Vereinbarung soll über das öffentliche Netz übertragen werden.

Das ist ein Angriffspunkt für den Lauscher: Er kann sich diese Vereinbarung verschaffen und damit die Nachricht entschlüsseln. Es sei denn, man überträgt nur solche Teile der Vereinbarung, die zwar die Verschlüsselung ermöglichen, nicht aber die Entschlüsselung.

Das heißt: Der Schlüssel E ist – wie die Nachricht auch – *öffentlich*. Er wird dem Sender der Nachricht über das öffentliche Netz zur Verfügung gestellt. Sicherheit vor Lauschangriffen ist nur dann gewährleistet, wenn es praktisch unmöglich ist, aus der verschlüsselten Nachricht y und dem Schlüssel E den *privaten* Entschlüsselungsschlüssel D zu rekonstruieren. Public-Key-Kryptographie beschäftigt sich mit Verfahren, die genau das leisten.

Konstruktion von privatem und öffentlichem Schlüssel

Ausgangspunkt ist eine superansteigende Folge von Zahlen $A = (a_1, a_2, \ldots, a_l)$. Superansteigend meint, dass jede Zahl der Folge größer als die Summe der Vorgänger ist. Die nebenstehende Tabelle zeigt ein Beispiel. Zu dieser Folge wird als *Modul* eine Zahl m bestimmt, die größer als die Summe aller Zahlen des privaten Schlüssels ist. Die linke Spalte der folgenden Tabelle zeigt den privaten Schlüssel unseres Beispiels. Ein dazu passendes m wäre $m = 614757665$.

A	*B*
3	37037034
5	61728390
9	111111102
18	222222204
36	444444408
73	286476829
147	585299336
293	543495329
586	472232993
1173	342053999
2345	57004655
4691	126354988
9380	228018620
18761	468382918
37522	322008171
75043	16912999
150088	58517354
300174	92343352
600349	197032382
1200699	406410442
2401397	185717541
4802794	371435082
9605589	140458177
19211177	268570676
38422355	549487030
76844708	459525039
153689416	304292413
307378833	6172839

Dann wird eine nicht zu kleine Zahl n gewählt, die zu m teilerfremd ist; in unserem Beispiel ist n =12345678. Wie wir im letzten Abschnitt gesehen haben, liefert der erweiterte euklidische Algorithmus eine zu n inverse Zahl modulo m:

$$(n \cdot n^{-1}) \bmod m = 1.$$

Wir erhalten $n^{-1} = 381909562$. Der private Schlüssel besteht aus der superansteigenden Zahlenfolge A und den Zahlen n und m:

$$(A, n, m).$$

Die Zahlen B des öffentlichen Schlüssels werden aus dem privaten Schlüssel abgeleitet. Sie sind definiert durch

$$b_i = (a_i \cdot n) \bmod m.$$

Der öffentliche Schlüssel besteht nur aus dieser Zahlenfolge.

Die nebenstehende Tabelle zeigt die Zahlenfolgen des privaten und des öffentlichen Schlüssels mit je 28 Dezimalzahlen.

Codierung

Der Absender der Nachricht kennt den öffentlichen Schlüssel. Er verschlüsselt seine Nachricht in folgenden Schritten:

1. Er bildet nach einem allgemein bekannten Verfahren aus seiner Nachricht eine Bitfolge, beispielsweise mittels internationalem 7-Bit-Code.
2. Dann zerlegt er seine Nachrichten in aufeinanderfolgende Abschnitte aus je l Bits: x_1, $x_{2, \ldots,} x_l$. In unserem Fall ist $l = 28$. In diesem Fall lassen sich je Abschnitt 4 Zeichen des internationalen 7-Bit-Codes darstellen. Beispielsweise führt das Wort „Nase“ auf die Bitfolge 1001110 1100001 1110011 1100101
3. Jede dieser Bitfolgen wandelt er in eine Dezimalzahl, indem er die Bits in aufsteigender Folge den Zahlen des öffentlichen Schlüssels zuordnet und alle diejenigen Zahlen aufaddiert, denen eine Eins zugeordnet ist. So erhält er als Codewort für den Abschnitt die Zahl $c = \sum_{k=1}^{l} b_k x_k$. Im Beispiel ergibt sich als Codewort die Zahl 4441449359.
4. Die Folge solcher Codewörter c übermittelt der Sender an den Empfänger der Nachricht.

Problem des Lauschers

Der Lauscher steht bei seinem Angriff vor einem Problem: Er kennt das Codewort c und die Zahlenfolge $B = (b_1, b_2, \ldots, b_l)$ des öffentlichen Schlüssels. Ihn interessiert aber die ursprüngliche Bitfolge $x = (x_1, x_2, \ldots, x_l)$. Er will also wissen, welche Zahlen der Folge B ausgewählt werden müssen, sodass sich die Summe c ergibt: $c = \sum_{k=1}^{l} b_k x_k$. Er steht vor demselben Prob-

lem wie jemand, der einen Rucksack gegebener Kapazität durch Auswahl von Gepäckstücken möglichst voll machen soll. Dieses *Rucksackproblem* (Knapsack Problem) ist bekanntlich schwer zu lösen, wenn die Anzahl der Gepäckstücke groß und deren Gewichte so verteilt sind, dass sich keine nützlichen Hinweise für den Packungsplan aus der Gewichtsverteilung herleiten lassen.

Decodierung

Allein die Kenntnis des öffentlichen Schlüssels und der Codewortfolge liefert keinen einfachen Weg zur Entschlüsselung der Nachricht. Der Empfänger aber besitzt den privaten Schlüssel. Ihm ist es ein Leichtes, die ursprüngliche Nachricht wieder sichtbar zu machen.

Die Entschlüsselung eines Codeworts c mit dem privaten Schlüssel (A, n, m) funktioniert so:

Mittels Modulo-Operation wird zum Codewort c das „private Äquivalent“ c' geschaffen:

$$c' = (c \cdot n^{-1}) \bmod m.$$

Es ist $c' = (\sum_{k=1}^{l} ((b_k \cdot n^{-1}) \bmod m) \cdot x_k) \bmod m$. Das folgt aus den Regeln (1) und (2) für die Addition und Multiplikation modulo m. Es gilt $(b_k \cdot n^{-1}) \bmod m = (a_k \cdot n \cdot n^{-1}) \bmod m = (a_k \cdot 1) \bmod m = a_k$. Daraus folgt

$$c' = \sum_{k=1}^{l} a_k x_k \ .$$

Nun muss der Empfänger das *Rucksackproblem* lösen: Er muss die Zahlen der Folge A so auswählen, so dass sich die Summe c' ergibt. Da es sich bei A um eine superansteigende Zahlenfolge handelt, ist dieses Problem mit linearem Aufwand unter Einsatz eines *gierigen Algorithmus* (GreedyAlgorithm) leicht zu lösen.

Im Beispiel ergibt sich für c' der Wert 402516341. Davon ziehen wir die größte der verfügbaren Zahlen, nämlich a_{28} = 307378833 ab. So verbleibt der Wert 95137610. Damit haben wir das letzte Bit der ursprünglichen Nachricht, eine Eins. Der verbleibende Wert ist zu klein, um a_{27} abziehen zu können. Das liefert eine 0 für das vorletzte Bit der Nachricht usw. Letztlich ergibt sich die ursprünglich gesendete Bitfolge.

Kritik

Das Merkle-Hellman-Kryptosystem ist nicht sicher. Adi Shamir hat einen Algorithmus vorgestellt, der unter Zuhilfenahme des öffentlichen Schlüssels die ursprüngliche Nachricht erzeugt. Dennoch ist es sinnvoll, sich mit dem Merkle-Hellman-Kryptosystem auseinanderzusetzen: Es ist einfach und zeigt, mit welcher Mathematik und mit welchen Rechenproblemen man es bei Kryptosystemen ganz allgemein zu tun bekommt: Modulo-Oprerationen, Algebra, Rechnen mit sehr großen ganzen Zahlen. Außerdem liefert es gute Beispiele für das Rucksackproblem: Das Rucksackproblem ist schwer lösbar für den öffentlichen und leicht lösbar für den privaten Schlüssel.

Chinesischer Restsatz

Der Satz

Gegeben seien die zueinander teilerfremden Zahlen m_1 und m_2. Es gilt also ggT(m_1, m_2) = 1. Ferner seien a_1 und a_2 Zahlen, die die Bedingungen $0 \le a_1 < m_1$ und $0 \le a_2 < m_2$ erfüllen. Dann gibt es eine ganze Zahl x, die das Gleichungssystem

$$x \bmod m_1 = a_1, \tag{3}$$

$$x \bmod m_2 = a_2. \tag{3'}$$

erfüllt.

Sei $m = m_1 \cdot m_2$. Naheliegenderweise sind mit x auch sämtliche Zahlen der Form $x + km$ (mit ganzen Zahlen k) Lösungen des Gleichungssystems, also sämtliche Zahlen der Restklasse modulo m. Diese Restklasse ist eindeutig bestimmt (Eindeutigkeit der Lösung modulo m). Für zwei Lösungen x und x' des Gleichungssystems gilt also $\overline{x} = \overline{x'}$. Der Überstrich kennzeichnet die Bildung des Restes modulo m.

Der Satz lässt sich auf Systeme mit einer beliebigen Anzahl von Gleichungen erweitern. Diese Allgemeinheit ist hier nicht vonnöten. Bei der Formulierung der Beweisschritte wird jedoch diese Verallgemeinerung im Blick behalten.

Konstruktiver Existenzbeweis

Wir suchen eine Zahl x, die bezüglich dem Modul m_1 den Rest a_1 und bezüglich dem Modul m_2 den Rest a_2 besitzt. Wir probieren den Ansatz

$$x = a_1 \cdot z_1 + a_2 \cdot z_2.$$

Von der Zahl z_1 wird verlangt, dass ihr Rest modulo m_1 gleich 1 und ihr Restmodulo m_2 gleich 0 ist. Bei der Zahl z_2 wird umgekehrt verlangt, dass $z_2 \bmod m_1 = 0$ und $z_2 \bmod m_2 = 1$.

Ich setze (allein aus Gründen der leichteren Verallgemeinerung des Satzes auf mehr als zwei Gleichungen) $M_1 = m/m_1 = m_2$ und $M_2 = m/m_2 = m_1$.

Da m_1 und M_1 teilerfremd sind, liefert uns der erweiterte euklidische Algorithmus zwei Zahlen b_1 und B_1, sodass

$$M_1 \cdot B_1 + m_1 \cdot b_1 = 1.$$

Entsprechend erhalten wir b_2 und B_2 derart, dass

$$M_2 \cdot B_2 + m_2 \cdot b_2 = 1.$$

Im vorliegenden Fall mit nur zwei Gleichungen ist $B_2 = b_1$ und $b_2 = B_1$. Die Zahlen

$$z_1 = M_1 \cdot B_1$$

und

$$z_2 = M_2 \cdot B_2.$$

besitzen die gewünschten Eigenschaften. Damit haben wir eine geeignete Zahl x konstruiert:

$$x = a_1 \cdot M_1 \cdot B_1 + a_2 \cdot M_2 \cdot B_2;$$

x ist eine Lösung des Gleichungssystems (3)-(3′).

Beispiel

Zur Illustration der Beweismethode konstruiere ich die Lösung für ein kleines (ausgedachtes) Problem der Himmelsmechanik.

Aufgabenstellung: Ein Planet wird von zwei Monden umrundet. Der erste braucht für die Umrundung 31 Tage, der zweite 65 Tage, immer relativ zur Sonnenposition gerechnet. Zurzeit sind vom Vollmond des ersten Mondes 7 Tage vergangen und der zweite steht bei Tag 25 seit seinem letzten Vollmond. Unsere Zeitrechnung beginnt, wenn beide Monde im Vollmond stehen. An welchem Tag ist die beschriebene Konstellation erreicht.

Lösung: Mit den Bezeichnungen des Satzes und des Beweises haben wir $a_1 = 7$, $m_1 = M_2 = 31$, $a_2 = 25$, $m_2 = M_1 = 65$. Daraus folgt

$$m = m_1 \cdot m_2 = 2015.$$

Wir sehen: Jede Mondkonstellation wiederholt sich nach 2015 Tagen. Der chinesische Restsatz liefert $B_1 = 65$ und $B_2 = 21$. Daraus folgt

$$x = 7\cdot65\cdot(-10) + 25\cdot31\cdot21 = 11725.$$

Wir erhalten den Rest modulo m zu $\bar{x} = 1655$. Das ist die hier bevorzugte Lösung des Gleichungssystems (3)-(3′). Seit dem letzten Mal, als beide Monde im Vollmond standen, sind 1655 Tage vergangen.

Beweis der Eindeutigkeit der Lösung

Sowohl x als auch x' seien Lösungen des Gleichungssystems (3)-(3′). Daraus folgt, dass die Differenz dieser Werte sowohl m_1 als auch m_2 als Teiler enthält: $(x\text{-}x') \bmod m_1 = 0$ und $(x\text{-}x') \bmod m_2 = 0$. Daher ist $x\text{-}x' = k_1\cdot m_1 = k_2\cdot m_2$ mit geeignet gewählten Werte für k_1 und k_2 (s. oben). Da aber $ggT(m_1, m_2) = 1$ ist, gibt es zwei Zahlen b_1 und b_2, sodass $1= m_1\cdot b_1 + m_2\cdot b_2$. Nach Multiplikation beider Seiten der Gleichung mit k_1 erhalten wir $k_1 = k_1\cdot m_1\cdot b_1 + k_1\cdot m_2\cdot b_2 = k_2\cdot m_2\cdot b_1 + k_1\cdot m_2\cdot b_2 = m_2\cdot(k_2\cdot b_1 + k_1\cdot b_2) = k\cdot m_2$ mit $k = k_2\cdot b_1 + k_1\cdot b_2$. Das bedeutet, dass $x\text{-}x' = k\cdot m_1\cdot m_2 = k\cdot m$. Daraus folgt die zu beweisende Gleichung $\bar{x} = \overline{x'}$.

Die eulersche Funktion φ

Die *eulersche Funktion* φ ist folgendermaßen definiert: Der Funktionswert $\varphi(m)$ ist gleich der Anzahl der Reste modulo m, die zu m teilerfremd sind:

$$\varphi(m) = |\{r \mid 0 \le r<m, ggT(r, m) = 1\}|.$$

Wir suchen eine einfache Formel für diese Funktion. Ist m das Produkt zweier zueinander teilerfremder Zahlen m_1 und m_2, ist also $ggT(m_1, m_2) = 1$, so gilt

$$\varphi(m) = \varphi(m_1\cdot m_2) = \varphi(m_1)\cdot\varphi(m_2).$$

Diese Formel nach dem Teile-und-herrsche-Prinzip bringt uns ein Stück voran. Sie ergibt sich folgendermaßen: Zu vorgegebenem x mit $0\le x<m$ finden wir Zahlen $a_1 = x \bmod m_1$ und $b_2 = x \bmod m_2$. Umgekehrt gibt es nach dem chinesischen Restsatz zu jedem a_1 mit $0 \le a_1<m$ und jedem a_2 mit $0 \le a_2<m$ genau ein x modulo m, das diese Gleichungen erfüllt. Zum Beweis der Teile-und-herrsche-Formel brauchen wir nur noch zu zeigen, dass folgende Äquivalenz gilt:

$ggT(x, m)=1$ gilt genau dann, wenn $ggT(a_1, m_1)=ggT(a_2, m_2)=1$.

Denn die Menge aller Paare (a_1, a_2) mit $ggT(a_1, m_1)=ggT(a_2, m_2)=1$ und $0\le a_1<m_1$ bzw. $0\le a_2<m_2$ hat aufgrund dieser Äquivalenzaussage und aufgrund der umkehrbar eindeutigen Zuordnung der (a_1, a_2)-Paare zu den x-Werten dieselbe Mächtigkeit wie die Menge aller x mit $ggT(x, m)=1$ und $0\le x<m$.

Beweis der Äquivalenzaussage: Vorbereitend zeige ich, dass $ggT(x, m)=1$ genau dann gilt, wenn $ggT(x, m_1) = ggT(x, m_2) = 1$. Ich starte mit der Annahme, dass $ggT(x, m) = 1$. Nehmen wir an, die Zahl c mit $1<c$ teilt m_1 und x, dann wäre c auch ein Teiler vom $m = m_1\cdot m_2$ im Widerspruch zu $ggT(x, m)=1$. Also muss $ggT(x, m_1)=1$ sein. Dementsprechend folgt $ggT(x, m_2)=1$. Nun zur Umkehrung: Vorausgesetzt wird $ggT(x, m_1) = ggT(x, m_2) = 1$ und wir setzen $ggT(x, m)=c$. Die Voraussetzungen ziehen $ggT(c, m_1) = ggT(c, m_2) = 1$ nach sich. Wegen $ggT(c, m_1)=1$ liefert der erweiterte euklidische Algorithmus Werte s und t derart, dass $tc + sm_1 = 1$. Multiplikation der Gleichung mit m_2 liefert die Gleichung $m_2tc + sm = m_2$. Die linke Seite der Gleichung besitzt den Teiler c, also auch die rechte. Also ist c Teiler von m_2 und damit ein gemeinsamer Teiler von x und m_2. Da aber $ggT(x, m_2) = 1$, muss c den Wert 1 haben. Soweit die Vorbereitung. Nun wird gezeigt, dass $ggT(x, m_1)=1$ genau dann gilt, wenn $ggT(a_1, m_1)=1$. Wegen $a_1 = x \bmod m_1$ erhalten wir mit einem passenden Wert k die Gleichung $x= km_1 + a_1$. Daraus folgt, dass jeder gemeinsame Teiler von x und m_1 auch ein Teiler von a_1 sein muss; umgekehrt muss jeder gemeinsame Teiler von x und a_1 auch ein Teiler von m_1

sein. Dasselbe gilt für die Ausdrücke mit dem Index 2. Letztlich ergibt sich die zu beweisende Äquivalenzaussage.

Die Teile-und-herrsche-Formel liefert uns die Möglichkeit, die eulersche Funktion immer weiter in Produkte zu zerlegen bis alle Argumente Primzahlen oder Primzahlpotenzen sind.

Nehmen wir eine Primzahl p. Da alle in Frage kommenden Zahlen, nämlich 1, 2, …, p-1, aufgrund der Primzahleigenschaft zu p teilerfremd sind, muss gelten

$$\varphi(p) = p-1. \tag{4}$$

Für den Fall, dass m das Produkt der voneinander verschiedenen Primzahlen p_1 und p_2 ist, folgt

$$\varphi(m) = \varphi(p_1 \cdot p_2) = \varphi(p_1) \cdot \varphi(p_2) = (p_1 - 1)\cdot(p_2 - 1). \tag{4'}$$

Für die allgemeine Formel zur Berechnung der eulerschen Funktion wird noch der Wert von $\varphi(p^n)$ für Primzahlpotenzen gebraucht. Deshalb soll auch dieser Wert hier noch bestimmt werden. Alle in Frage kommenden Restezahlen sind 1, 2, …, p^n-1. Das sind p^n-1 Zahlen. Aber dabei sind auch Zahlen, die zu p^n nicht teilerfremd sind. Das sind alle Zahlen der Form pk mit $0<pk<p^n$ bzw. $0<k<p^{n-1}$. Das sind p^{n-1}-1 Zahlen. Also bleiben p^n-1-(p^{n-1}-1) oder $p^{n-1}(p-1)$ zu p^n teilerfremde Zahlen übrig. Das heißt

$$\varphi(p^n) = p^{n-1}(p-1).` \tag{4''}$$

Die Sätze von Pierre de Fermat und Leonhard Euler

Die Gruppe der zum Modul teilerfremden Reste

Wir rechnen in diesem Abschnitt durchweg mit den Resten modulo m mit dem frei aber fest gewählten Modul m, $1<m$. Zur Entlastung des Schriftbildes lasse ich die Überstreichungen weg. Also: Anstelle $\overline{\overline{a}+\overline{b}}$ für die Summe von Resten steht hier einfach $a + b$; und für das Produkt $\overline{\overline{a}\cdot\overline{b}}$ von Resten schreibe ich $a \cdot b$. Die Menge der Reste ist $R = \{0, 1, 2, \ldots, m\text{-}1\}$.

Wir konzentrieren uns in diesem Abschnitt auf die Produkte von Resten. Auch den Multiplikationspunkt lasse ich – wie allgemein üblich – meist weg. Für a, b, $c \in R$ gelten diese Gesetze:

$ab \in R$, Abgeschlossenheit

$ab = ba$, Kommutativgesetz

$(ab)c = a(bc)$. Assoziativgesetz

Sind das Gesetz der Abgeschlossenheit und das Assoziativgesetz erfüllt, spricht man von einer *Halbgruppe.* Sind für eine mathematische Struktur alle diese Gesetze (Axiome) erfüllt, handelt es sich um eine *abelsche* Halbgruppe. Von *abelschen Gruppen G* verlangt man darüber hinaus die Existenz der Eins und der inversen Elemente.

Für jedes $a \in G$ gilt

$a \cdot 1 = a$. Existenz des Einselements

Zu jedem Element a gibt es ein Element a^{-1}, sodass

$aa^{-1} = 1$. Existenz des inversen Elements

R ist keine abelsche Gruppe: Zwar haben wir eine Eins, aber nicht zu jedem Element gibt es ein inverses. Selbst wenn wir für den Modul m eine Primzahl nehmen: Die Null gehört zu den Resten und dieses Element besitzt kein inverses. Leicht einzusehen ist, dass es in abelschen Gruppen nur eine Eins und zu jedem Element auch nur ein inverses Element geben kann.

Wir schränken jetzt die Menge der Reste so ein, dass sämtliche Gruppengesetze gelten:

$$G = \{ x | \, x \in R, \, \text{ggT}(x, m) = 1 \}.$$

Das heißt: Die Menge G enthält nur noch die Reste, die zum Modul m teilerfremd sind. Damit wissen wir aber auch schon, wie viele Elemente diese Menge enthält:

$$|G| = \varphi(m).$$

Nun müssen wir noch zeigen, dass die Gruppengesetze erfüllt sind. Kommutativ- und Assoziativgesetz machen keine Schwierigkeiten. Und da ggT(1, m) = 1, haben wir auch das Einselement hinübergerettet. Aber wie steht es mit der Abgeschlossenheit und der Existenz des inversen Elements? Hier sind Beweise nötig.

Seien a, $b \in G$ und ggT(ab, m)=c. Wegen ggT(b, m)=1 liefert uns der erweiterte euklidische Algorithmus zwei Zahlen s und t derart, dass $bs + mt = 1$. Multiplikation mit a ergibt $abs + amt = a$. Die linke Seite der Gleichung besitzt den Teiler c. Also muss c Teiler von a sein. Da aber voraussetzungsgemäß ggT(a, m)=1 ist, muss c gleich 1 sein. Demnach gilt: ggT(ab, m)=1 und folglich ist $ab \in G$. Das beweist: G ist bezüglich der Multiplikation abgeschlossen.

Für jedes Element $a \in G$ ist ggT(a, m)=1. Demnach hat a zumindest in R ein inverses Element a^{-1}, das sich mit dem erweiterten euklidischen Algorithmus berechnen lässt. Wegen $aa^{-1} = 1$ kann dieses Element mit m keinen gemeinsamen Teiler größer 1 haben. Also gehört a^{-1} zu G.

Insgesamt haben wir den wichtigen Satz: Die Menge der zu m teilerfremden Reste bildet bezüglich der Multiplikation eine abelsche Gruppe.

Beispiel: Sei $m = 9$. $R = \{0, 1, 2, \ldots, 8\}$. $\varphi(m) = \varphi(9) = \varphi(3^2) = 3^1 \cdot (3\text{-}1) = 6$. $G = \{1, 2, 4, 5, 7, 8\}$. Die Verknüpfungstabelle veranschaulicht Abgeschlossenheit und Inversenbildung.

·	1	2	4	5	7	8
1	1	2	4	5	7	8
2	2	4	8	1	5	7
4	4	8	7	2	1	5
5	5	1	2	7	8	4
7	7	5	1	8	4	2
8	8	7	5	4	2	1

Untergruppen und Restklassen

Wir bleiben bei der Gruppe G der zum Modul m teilerfremden Reste. Wir entnehmen der Gruppe ein beliebiges Element a und bilden die Potenzen a^i dieses Elements. Diese Potenzen liegen alle in der endlichen Menge G und nur endlich viele dieser Potenzen können voneinander verschieden sein. Folglich muss es zwei Exponenten i und j geben, so dass $a^i = a^j$. Also ist $a^{i-j} = 1$, wobei ohne Beschränkung der Allgemeinheit $j \leq i$ angenommen werden kann. Es gibt also eine nichtnegative Zahl k derart, dass $a^k = 1$. Sei k die kleinstmögliche dieser Zahlen. Die Menge sämtlicher Potenzen von a wird mit $\langle a \rangle$ bezeichnet und ist gegeben durch

$$\langle a \rangle = \{1, a, a^2, \ldots, a^{k-1}\}.$$

Es ist leicht einzusehen, dass $\langle a \rangle$ eine abelsche Gruppe mit k verschiedenen Elementen ist. Es handelt sich um eine Untergruppe von G. Wir bezeichnen diese fürs Folgende mit U:

$$U = \langle a \rangle.$$

Nun wählen wir ein Element $b \in G$. Die Menge der Elemente bx mit $x \in U$ wird mit bU bezeichnet. Wir bilden alle möglichen dieser *Nebenklassen* und stellen fest: Je zwei dieser Nebenklassen sind entweder identisch oder sie haben kein Element gemeinsam. Sei nämlich beispielsweise $x \in aU \cap bU$. Dann ist $x = au = bv$ mit $u, v \in U$. Wegen $b = auv^{-1}$ ist demnach $b \in aU$ und folglich $bU \subset aU$. Die Umkehrung $aU \subset bU$ lässt sich ebenso zeigen. Also muss $aU = bU$ sein.

Sämtliche Nebenklassen bilden eine Zerlegung der Gesamtmenge G. Da die Nebenklassen alle dieselbe Mächtigkeit $|aU| = |U| = k$ besitzen, muss diese Mächtigkeit ein Teiler von $|G|$ sein. Es gilt

$$k \cdot f = \varphi(m)$$

mit einem geeignet gewählten Faktor f. Daraus ergibt sich $a^{\varphi(m)} = a^{kf} = (a^k)^f = 1^f = 1$. Damit haben wir den für die Public-Key-Kryptographie wichtigen *Satz von Euler:* Es ist

$$a^{\varphi(m)} \bmod m = 1 \tag{6}$$

für alle ganzen Zahlen a mit ggT(a, m)=1.

Ist p eine Primzahl, dann geht diese Formel in den *kleinen Satz von Fermat* über:

$$a^{p-1} \bmod p = 1. \tag{6'}$$

Das RSA-Kryptosystem

Der RSA Algorithmus wurde 1978 von Ron Rivest, Adi Shamir und Leonard Adleman vom Massachusetts Institute of Technology (MIT) veröffentlicht. Sie setzten damit die Arbeiten von Whitfield Diffie und Martin Hellman zur Public-Key-Kryptografie fort.

Der Empfänger der Nachricht konstruiert seinen privaten Schlüssel folgendermaßen: Er wählt zwei ziemlich große Primzahlen p und q und behält diese für sich. Dem Sender der Nachricht übermittelt er als öffentlichen Schlüssel ein Zahlenpaar (E, m). Die Zahl m ist das Produkt der beiden Primzahlen

$$m = pq$$

und E ist eine weitere Zahl, die der Sender weitgehend nach dem Zufallsprinzip gewählt hat, mit einer Einschränkung: Die Zahl muss teilerfremd zu $\varphi(m) = (p-1)\cdot(q-1)$ sein, s. Gleichung (4′).

Der Absender wandelt seine Nachricht nach einem allseits bekannten Verfahren in eine Folge nichtnegativer ganzer Zahlen. Sei X eine dieser Zahlen. Vorausgesetzt wird, dass $0 \leq X < m$ ist. Zu jeder dieser Zahlen X bildet der Sender ein Codewort C nach der Formel

$$C = X^E \bmod m.$$

Dieses Wort erhält der Empfänger der Nachricht. Sowohl dem unbefugten Lauscher als auch dem Empfänger der Nachricht stellt sich die Aufgabe, aus dem C die ursprüngliche Nachricht X zurück zu gewinnen. Dem Lauscher steht dafür nur der öffentliche Schlüssel (E, m) zur Verfügung. Aber damit kann er nicht viel anfangen. Ihm fehlt das Herrschaftswissen des Empfänger, nämlich die Kenntnis der Primzahlen p und q. Er könnte versuchen, diese Primzahlen aus deren Produkt m zu gewinnen. Aber diese Primzahlzerlegung gehört für ausreichend große Zahlen zu den wirklich schweren und praktisch unlösbaren Problemen.

Wie nun nutzt der Empfänger sein Herrschaftswissen bei der Lösung der Dekodierungsaufgabe? Er macht auch eine Modulo-Rechnung. Aber er wählt als einen weiteren Modul den Wert $\varphi(m) = (p-1)\cdot(q-1)$ der eulerschen Funktion. Da E und $\varphi(m)$ teilerfremd sind, liefert der erweiterte euklidische Algorithmus einen Wert D, sodass $E \cdot D \bmod \varphi(m) = 1$. Mit einem geeigneten k gilt also

$$ED = k \cdot \varphi(m) + 1.$$

Vorausgesetzt, es ist ggT(X, m)=1. Dann ist der eulersche Satz (5) anwendbar. Er liefert

$$X^{\varphi(m)} \bmod m = 1. \tag{7}$$

Nun zur Dekodierung: Der Empfänger berechnet $C^D \bmod m$ und erhält

$$C^D \bmod m = (X^E \bmod m)^D \bmod m = X^{ED} \bmod m.$$

Ein paar weitere Umformungen liefern – unter Berücksichtigung von $X<m$ – das gewünschte Resultat:

$$X^{ED} \bmod m = X^{k\cdot\varphi(m)+1} \bmod m = ((X^{\varphi(m)})^k \cdot X) \bmod m = ((X^{\varphi(m)} \bmod m)^k \cdot X) \bmod m$$
$$= 1^k \cdot X \bmod m = X \bmod m = X.$$

Damit ist leider noch nicht alles erledigt: Wir müssen noch zeigen, dass die Dekodierung auch dann noch funktioniert, wenn entweder p oder q ein Teiler von X oder wenn $X = 0$ ist. Für $X = 0$ ist $C = 0$ und die Dekodierung ergibt wieder den Wert 0. Dieser Fall wird also trivialerweise korrekt kodiert und dekodiert.

Werde nun X von einer der beiden Primzahlen geteilt. Nehmen wir beispielsweise an, dass q Teiler von X ist: $X = qY$. Wegen $X < m = pq$ kann p nicht Teiler von Y sein. Teiler von p ist q ebenfalls nicht, da es sich um verschiedene Primzahlen handelt. Mit dem erweiterten euklidischen Algorithmus ergibt sich dann, dass p auch nicht Teiler von X sein kann. Also gilt $\mathrm{ggT}(X, p)=1$ und damit $X^{p-1} \bmod p = 1$. Da p-1 Teiler von $\varphi(m)$ ist, können wir mit einem k' schreiben $ED = k' \cdot (p-1) + 1$. Analog zur oben durchgeführten Herleitung ergibt sich $X^{ED} \bmod p = X \bmod p$. Diesmal können wir auf der rechten Seite die Modulo-Bildung nicht weglassen, da nicht sichergestellt ist, dass X kleiner als p ist. Es ist bequem, die Gleichung etwas umzustellen:

$$(X^{ED} - X) \bmod p = 0.$$

Außerdem gilt

$$(X^{ED} - X) \bmod q = 0,$$

denn X enthält den Faktor q, was zur Folge hat, dass $X \bmod q = 0$. Der chinesische Restsatz liefert schließlich $(X^{ED} - X) \bmod m = 0$, was wir wieder in die gewünschte Form bringen:

$$X^{ED} \bmod m = X \bmod m = X.$$

Die Entschlüsselung funktioniert demnach für sämtliche Zahlen X im Bereich von 0 bis m-1.

Beispiel: Der Empfänger wählt die Primzahlen p=7 und q=11. Daraus erhält er den Wert der eulerschen Funktion $\varphi(m)= \varphi(pq)= (p-1)(q-1) = 60$. Der öffentliche Modul m ergibt sich zu $pq = 77$. Eine zu $\varphi(m)$ teilerfremde Zahl ist $E = 7$. Der öffentliche Codierungsschlüssel ist demnach gleich $(E, n) = (7, 77)$. Der Sender will als Nachricht die Zahl $X = 5$ senden. Er verschlüsselt diese mithilfe der Modulo-Potenzierung:

$$5^7 \bmod 77 = \overline{5^7} = 47.$$

Das Codewort ist also gleich $C = 47$. Dieses wird an den Empfänger der Nachricht übermittelt. Der Empfänger besorgt sich die Dekodierzahl D mittels euklidischem Algorithmus und erhält $D = 43$. Zur Ermittlung der ursprünglichen Nachricht muss er die Formel

$$C^D \bmod m = 47^{43} \bmod 77 = \overline{47^{43}}$$

auswerten und erhält $X = 5$.

Die Zahlen, die in diesen Formeln auftreten, können riesig werden. Man halte sich nur vor Augen, dass es sich in der Praxis nicht um ein-, zweistellige Zahlen, sondern um Zahlen mit hunderten von Stellen handelt. Und die Potenzierung großer Zahlen mit großen Zahlen macht die Sache nicht gerade einfach. Aber man kann dafür sorgen, dass die Zahlen immer halbwegs handlich bleiben, da

X	C	X'
0	0	0
1	1	1
2	51	2
3	31	3
4	60	4
5	47	5
6	41	6
7	28	7
8	57	8
9	37	9
10	10	10
11	11	11
12	12	12
13	62	13
14	42	14
15	71	15
16	58	16

sich die Operanden einer jeden Operation und auch das Ergebnis derselben durch die Modulo-Bildung stets auf Werte kleiner als m reduzieren lassen.

Die Tabelle zeigt zu einigen Zahlen X die zugehörigen Codezahlen C und das Ergebnis X' der Rückwandlung durch den Empfänger.