Kryptographie

Vorlesung, zuerst gehalten im Wintersemester 2013/14

Tomas Sauer

Version 0.0
Letzte Änderung: 25.1.2014

Statt einer Leerseite . . . | 0

Chaos is found in greatest abundance whereever order is being sought. It always defeats order, because it is better organized.

T. Pratchett, *Interesting times*

Die wahren Analphabeten sind schließlich diejenigen, die zwar lesen können, es aber nicht tun. Weil sie gerade fernsehen.

L. Volkert, *SZ–Online, 11.7.2009*

When the epoch of analogue (which was to say also the richness of language, of *analogy*) was giving way to the digital era, the final victory of the numerate over the literate.

S. Rushdie, *Fury*

The most incredible thing about miracles is that they happen.

G. K. Chesterton, *The Innocence of Father Brown*

And it didn't stop being magic just because you found out how it was done.

T. Pratchett, *Wee Free Men*

Tomas Sauer
Lehrstuhl für Mathematik mit Schwerpunkt Digitale Bildverabeitung
Universität Passau
Innstr. 43
94032 Passau

0 Inhaltsverzeichnis

These characters, as any one might readily guess, form a cipher that is to say, they convey a meaning; but then, from what is known of Kidd, I could not suppose him capable of constructing any of the more abstruse cryptographs

Edgar Allan Poe, *The Gold Bug*

1 Kryptographie und klassische Verfahren

Kryptographie ist die Wissenschaft, Nachrichten so zu verschlüsseln, daß sie nur mittels eines geeigneten Entschlüsselungsverfahrens wieder lesbar gemacht werden können.

1.1 Ein bisschen Definiererei

Es ist ein schöner Brauch in der Mathematik, daß man die Dinge, über die man reden will, erst einmal klar und deutlich definiert. Dies wird dann auch das **Modell** einer Nachricht und eines Kryptosystems festlegen, mit dem wir uns befassen wollen. Dabei folgt die Darstellung dem Buch (Wätjen, 2008), wenn auch nicht in jedem Detail.

Definition 1.1 (Alphabete & Räume)

1. *Ein **Alphabet** A ist eine endliche Menge und kann mit*[1] $\mathbb{Z}_m := \{0, \ldots, m-1\}$, $m = \#A$, *identifiziert werden.*

2. *Ein **Klartextraum**[2] ist eine höchstens abzählbare Menge $\mathscr{M}$. Die natürliche Wahl ist*
$$\mathscr{M} = A^* = \bigcup_{n \in \mathbb{N}} A^n.$$

3. *Ein **Schlüsselraum** $\mathscr{K}$ und ein **Chiffreraum** $\mathscr{C}$ sind ebenfalls höchstens abzählbare Mengen mit $\#\mathscr{C} \geq \#\mathscr{K}$.*

4. *Eine **Chiffriertransformation**[3] ist eine Abbildung $E : \mathscr{K} \times \mathscr{M} \to \mathscr{C}$, die jeder Nachricht $M \in \mathscr{M}$ und jedem Schlüssel $K \in \mathscr{K}$ den verschlüsselten Text $E(K, M)$*

[1] Zu dieser Menge wird es noch einiges zu sagen geben. Aber nicht jetzt.

[2] Man fragt sich allerdings, wo hier der Raum ist, denn eigentlich ist es eine strukturlose Menge.

[3] Der Name „Chiffre“ stammt übrigens vom Wort *Zephyrus*, mit dem Fibonacci in seinem „Liber Abaci“ die Ziffer Null bezeichnet, siehe (Sigler, 2002).

zuordnet. Die ***Dechiffriertransformation*** $D : \mathscr{K} \times \mathscr{C} \to \mathscr{M}$ *bestimmt dann wieder den Ausgangsnachricht dazu, die zentrale Forderung ist also, daß*

$$D(K, E(K, M)) = M, \qquad M \in \mathscr{M}, \tag{1.1}$$

gelten muss.

Übung 1.1 Zeigen Sie: Die Menge A^* ist für ein endliches Alphabet A abzählbar, die Menge

$$A^\infty = \{a = (a_0, a_1, \dots) \ : \ a_j \in A\}$$

aller *unendlichen* Folgen hingegen ist für alle Alphabete mit mindestens zwei Symbolen überabzählbar. ◊

Die wesentlichen Ziele eines guten Verschlüsselungsverfahrens sind dann:

1. Die Abbildungen D und E müssen effizient berechenbar sein[4].
2. Das System muss leicht benutzbar sein.
3. Die Sicherheit muss auf der Geheimhaltung des Schlüssels, nicht des Algorithmus basieren.
4. *Geheimhaltungsanforderung:* Es darf nicht möglich sein, den Schlüssel K aus der Kenntnis von M und $E(K, M)$ zu bestimmen. Das Verfahren ist also **nicht** symmetrisch in K und M.
5. *Authentizitätsforderung:* Es darf ohne Kenntnis von K nicht möglich sein, zu einem $M \in \mathscr{M}$ ein $C \in \mathscr{C}$ zu erzeugen, so daß $D(K, C) = M$ ist. Das Fälschen von verschlüsselten Nachrichten soll also praktisch unmöglich sein.

Das alles ist natürlich sehr abstrakt und in dieser Allgemeinheit kaum brauchbar. Deswegen sehen wir uns jetzt erst einmal ein paar klassische Verschlüsselungsverfahren an und übertragen diese in den Formalismus von Definition 1.1. Eine sehr schöne populärwissenschaftliche Einführung in das Thema mit vielen Beispielen findet sich in (Singh, 2000b).

1.2 Klassische Verfahren

Beispiel 1.2 (Skytale) *Eines der ältesten Verschlüsselunsverfahren ist die sogenannte* ***Skytale****, bei der ein Lederstreifen um einen konischen Stab gewickelt wird, siehe Abb. 1.1. Die Buchstabenreihenfolge auf dem abgewickelten Lederstreifen ist dann der verschlüsselte Text.*

Der Trick bei der Skytale besteht offenbar in einer Vertauschung der Buchstaben. Gehen wir der Einfachheit halber davon aus, daß unsere Skytale prismatisch mit n Seiten und von unendlicher Länge ist[5], dann haben wir es offenbar mit $\mathscr{M} =$

[4] Ob Echtzeit oder nicht hängt von der Anwendung und der benötigten Sicherheit ab.

[5] Ja, sowas kann nur ein Mathematiker für einfach halten ...

Abbildung 1.1: Abbildung einer Skytale. Quelle: Wikimedia Commons.

$\mathscr{C} = A^*$ für ein „normales“ Alphabet zu tun[6], Klartext- und Chiffreraum können also durchaus identisch sein. Der Schlüssel besteht nun in einer Vertauschung der Buchstaben und zwar in der folgenden Weise[7]:

$$\begin{bmatrix} a_0 & a_1 & \dots & a_{n-1} \\ a_n & a_{n+1} & \dots & a_{2n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{(m-1)n} & a_{(m-1)n+1} & \dots & a_{mn-1} \end{bmatrix} \to E\left(a_0, a_n, \dots, a_{(m-1)n}, a_1, \dots, a_{mn-1}\right).$$

Der Schlüssel K ist dabei die Zahl n und die Chiffriertransformation E bildet eine Nachricht $M = (a_1, \dots, a_N)$ auf

$$E(K, M) = (a_{k+jn} : j = 0, \dots, m, k \in \mathbb{Z}_n), \qquad m = \lfloor N/m \rfloor + 1,$$

ab, der Klartext wird also modulo n zerhackt. Ist n unbekannt, so ist die Chiffre erst einmal nicht so leicht zu knacken, mit ein bisschen Computerunterstützung kann man aber sehr leicht alle potentiellen Klartexte eines chiffrierten Codes erzeugen.

Übung 1.2 Schreiben Sie ein Programm (z.B. in `Matlab`), das eine Skytale simuliert (Chiffrierung und Dechiffrierung) und ein Programm, das versucht, diesen Code zu knacken, indem es alle potentiellen Klartexte erzeugt. ◊

Definition 1.3 (Transpositionschiffre) *Eine **Transpositionschiffre** für Wortlänge N besteht aus einer Permutation $f : \mathbb{Z}_N \to \mathbb{Z}_N$ und der Chiffriertransformation*

$$E(f, M) = \left(a_{f(j)} : j \in \mathbb{Z}_N\right), \qquad M \in A^N.$$

[6]Leerzeichen wurden mit einer Skytale normalerweise nicht übermittelt.

[7]Mathematiker lieben es, Indizierungen mit 0 zu beginnen. Und zwar vor allem deswegen, weil es oftmals Dinge vereinfacht.

Historisch basieren Transpositionschiffren oftmals auf geometrischen Schemata, da diese leicht zu merken und zu kommunizieren sind. Dabei sind diese Ansätze durchaus mächtig, denn da es ja $(N-1)!$ Permutationen einer Menge mit N Elementen gibt, macht der Brute–Force–Ansatz sehr schnell keinen Spaß mehr. Will man die Chiffre noch verbessern, so kann man beispielsweise auch Permutationen $f : \mathbb{Z}_N \to \mathbb{Z}_{N'}$ mit $N' \gg N$ betrachten und die leeren Zeichen zufällig auffüllen.

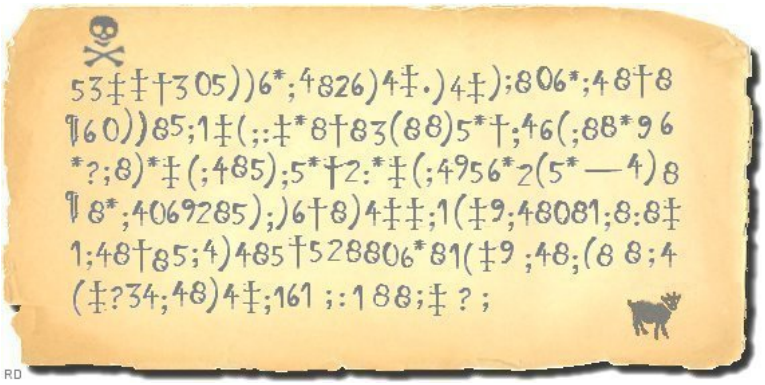


Abbildung 1.2: Die Chiffre aus (Poe, 1843). Ein Hinweis für angehende Kryptoanalytiker: Der Totenkopf und die Ziege haben nichts mit der Chiffre zu tun. Quelle: Dirk Rijmenants

Ein literarischer Klassiker der Chiffrierung ist die Kurzgeschichte *The Gold Bug* von Edgar Allan Poe (Poe, 1843), in der die Dechiffrierung der Chiffre aus Abb. 1.2 ausgesprochen lesenswert beschrieben wird[8]. Die hier verwendete Chiffre ist sehr einfach, nämlich eine Bijektion f zwischen dem Klartextalphabet A und einem Chiffrealphabet B mit

$$E(f, M) = (f(a_j) \ : \ j \in \mathbb{Z}_N), \qquad M \in \mathbb{Z}_N.$$

Das ist eine **einfache Substitutionschiffre**, jedes Zeichen wird auf eine wohldefinierte Art und Weise durch ein anderes ersetzt. Solche Chiffren kann man durch Häufigkeitsanalysen und Plausibilität recht einfach knacken, solange die Texte nur lang genug sind. Das wird auch in (Poe, 1843) sehr schön vorgeführt.

Will man es dem „Codeknacker" ein klein bisschen schwieriger machen, dann kann man manchen Symbolen auch mehrere Chiffren zuordnen.

Definition 1.4 (Homophone Substitutionschiffren) *Eine **homophone Substitutionschiffre** basiert auf zwei Alphabeten A, B mit $\#A \leq \#B$ und einer injektiven Abbildung*

$$f : A \to 2^B := \{B' \ : \ B' \subseteq B\},$$

[8]Die Hauptperson, Legrande, findet aufgrund dieses Hinweises dann auch den Schatz des Piraten Captain Kidd.

mit der Eigenschaft

$$f(a) \neq \emptyset, \quad a \in A \qquad \textit{und} \qquad f(a_1) \cap f(a_2) = \emptyset, \quad a_1 \neq a_2. \tag{1.2}$$

Formal ist dann $\mathscr{M} = A^*$, $\mathscr{C} = B^*$ *und* $\mathscr{K}$ *die Menge aller* $f : A \to 2^B$, *die* (1.2) *erfüllen. Die Chiffriertransformation bestimmt dann*

$$E(f, M) : (a_1, \ldots, a_N) \mapsto (b_1, \ldots, b_N), \qquad b_j \in f(a_j),$$

wobei die Auswahl von b_j *zufällig oder nach einer fest vorgegebenen Regel erfolgen kann.*

Der Vorteil dieser Methode ist, daß die normale Häufigkeitsanalyse versagt, indem man beispielsweise den Zeichen, die häufiger erscheinen[9] mehrere Substitutionen zuordnet.

Noch ein wenig besser wird es, wenn man auf der Basis eines Codeworts den Schlüssel in jedem Schritt ändert, was dann als **polyalphabetische Substitutionschiffre** bezeichnet wird. Die klassische Version davon, die **Vigenère–Chiffre** verwendet eine zyklische Verschiebung.

Definition 1.5 (Vigenère–Chiffre) *Für das Alphabet* $A = \mathbb{Z}_m$ *setzt man* $\mathscr{M} = \mathscr{K} = \mathscr{C} = A^N$ *und verwendet fr* $M = (a_0, \ldots, a_{N-1}) \in \mathscr{M}$ *und* $K = (k_0, \ldots, k_{N-1}) \in \mathscr{K}$ *die Chiffriertransformation*

$$E(K, M) = ((a_j + k_j)_m \ : \ j = 0, \ldots, N-1),$$

wobei $(a + b)_m$ *die Addition* ***modulo*** m *in* $\mathbb{Z}_m$ *bezeichnet, also*

$$(a + b)_m = \begin{cases} a + b, & a + b < m, \\ a + b - m, & a + b \geq m. \end{cases} \tag{1.3}$$

Ist der Schlüssel K *kürzer als die Nachricht* M, *so kann man ihn periodisch fotsetzen.*

Bemerkung 1.6 ($\mathbb{Z}_m$) *Nachdem man es ja doch irgendwann sagen muss:* $\mathbb{Z}_m$ *steht eigentlich für* $\mathbb{Z}/m\mathbb{Z}$, *also für die Menge aller bezüglich* m *ununterscheidbaren* ***Restklassen***. *Zwei Zahlen* $a, b \in \mathbb{Z}$ *gehören zu derselben Restklasse* $[a] = [b]$, *wenn* $a - b \in m\mathbb{Z}$, *also die Differenz ein Vielfaches von* m *ist. Die einfachsten* ***Repräsentanten*** *dieser Restklassen sind* $0, \ldots, m-1$, *denn offenbar ist*

$$\mathbb{Z}/m\mathbb{Z} = \bigcup_{j=0}^{m-1} [j],$$

jede andere Zahl fällt in eine dieser Restklassen[10]. *Und weil man nicht so viel schreiben will, kürzt man das einfach mit* $\mathbb{Z}_m = \{0, \ldots, m-1\}$ *ab.*

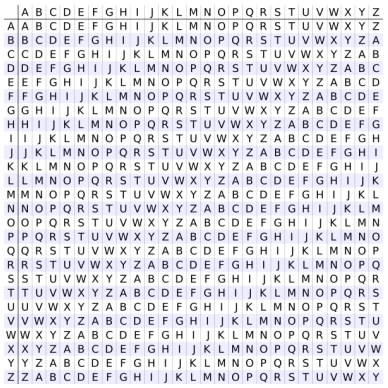
Mit Restklassen kann man rechnen, indem man $[a] \pm [b] = [a \pm b]$ *setzt*[11] *und mit unseren Repräsentanten von gerade eben erhalten wir eine zyklische Addition, die dann* $\mathbb{Z}_m \times \mathbb{Z}_m \to \mathbb{Z}_m$ *verarbeitet. Und die ist dann genau* (1.3).

Zur praktischen Durchführung dieser Verschlüsselung wählt man ein quadratisches Schema wie in Abb. 1.3, womit die Ver- und Entschlüsselung recht idiotensicher und effizient durchführbar wird.

[9] In der deutschen Sprache der Buchstabe „e“.

[10] Division mit Rest ...

[11] Und ja: Man kann auch multiplizieren, was formal auch nicht komplizierter ist



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Abbildung 1.3: Das „Vigenère–Quadrat“ zur Durchfhung der Verschlüsselung. In der Spalte wählt man den Buchstaben des Schlüssels, in der Zeile den Buchstaben der Nachricht.

> *In theory there is no difference between theory and practice. In practice there is.*
>
> Yogi Berra

2 Arithmetik & Algorithmen

Ein wesentlicher Punkt in der modernen Kryptographie sind Berechnungen, die man zwar durchführen kann, die aber so aufwendig sind, daß es sich de facto eben doch nicht geht, zumindest, wenn man die Lösung auch noch erleben will. Um das verstehen zu können, sehen wir uns in diesem Kapitel die Prinzipien der Darstellung von Zahlen und der zugehörigen Rechenoperationen ansehen – insbesondere natürlich die Frage, wie man diese Sachen *effizient* macht[12].

2.1 Ganze Zahlen

Fangen wir mit einer "flexiblen" Darstellung von ganzen Zahlen (fast) beliebiger Größe an und den effizienten Implementierungen der fundamentalen **Ganzzahloperationen**

- Addition/Subtraktion,
- Multiplikation,
- Division und Restbestimmung,

Die bestimmende Größe auf einem modernen Digitalcomputer ist das sogenannte **Wort**, das ist eine Gruppe von Bits, die der Rechner "auf einmal" verarbeiten kann. Die Anzahl der Bits in einem Wort bezeichnet man als **Wortlänge** w; in aktuellen[13], handelsüblichen Prozessoren beträgt momentan die Wortlänge normalerweise 64 Bit. In einem Wort kann man die Zahlen

$$\mathbb{Z}_B = \mathbb{Z}/\langle B\rangle = \{0, \ldots, B-1\}, \qquad B = 2^w,$$

darstellen. Da wir später mit richtig großen (Prim-)Zahlen arbeiten wollen, reicht natürlich ein Wort nicht aus, wir brauchen deutlich mehr.

Definition 2.1 (Multiprecision–Zahlen)

[12] Wenn wir wirklich behaupten wollen, eine Rechnung wäre zu komplex oder aufwendig, dann sollten wir nicht den dümmstmöglichen Algorithmus als Maßstab nehmen.

[13] Weil sich das dauernd ändert: Stand ist Oktober 2013.

1. *Eine ganze Zahl* p *der* ***Länge***[14] $N = \ell(p)$ *in der* ***Multiprecision–Arithmetik*** $\mathbb{M}_w$ *ist von der Form*

$$p = \pm \sum_{j=0}^{N} p_j B^j, \qquad p_j \in \mathbb{Z}_B, \quad 0 \le N < 2^{w-1}$$

und wird mit $N + 2$ *Worten dargestellt als*

$\frac{1-\operatorname{sgn} p}{2} 2^{w-1} + N$	p_0	$\cdots$	p_N

2. *Eine Zahl* $p \in \mathbb{M}_w$ *heißt* ***normal*** *oder* ***normalisiert****, wenn* $p_{\ell(p)} \neq 0$.

Bemerkung 2.2 *Diese Darstellung von "ganzen Zahlen* beliebiger *Größe" ist erst einmal ein bißchen verwunderlich, denn die Größe der darstellbaren Zahlen ist natürlich beschränkt, denn dadurch, daß die Anzahl der Ziffern im ersten Wort der Darstellung codiert ist und wir auch noch eine Binästelle für das Vorzeichen opfern, kann solch eine Zahl nur aus* $B/2 - 1$ *Worten bestehen.*

Trotzdem ist das in der Praxis keine Einschränkung, denn normalerweise wird auch der Speicher eines Computers ohnehin nur über ein Wort adressiert, das heißt, die Beschränkung bedeutet, daß eine Zahl maximal 50% des adressierbaren Hauptspeichers belegen darf. Und wenn man mal so eine Zahl hat, dann ist man sowieso in Schwierigkeiten. Außerdem ist das bereits auf einem "altmodischen" 32–Bit–Rechner schon eine ganze Menge: Hier besteht eine Multiprecision–Zahl aus maximal $2^{31} - 1$ *Zahlen zu je* $4 = 2^2$ *Bytes, also hat man mindestens*[15] $2^{33} \sim 8 \times 10^9$ *Bytes, also etwa 8GB pro Zahl zur Verfügung. Bei einem 64–Bit–Prozessor sind es dann schon mehr als* 10^{18} *Bytes pro Zahl und das passt nun wirklich nicht mehr in ein Smartphone.*

Beginnen wir mit der einfachsten Operation, nämlich der Addition[16], die natürlich komponentenweise und mit Übertrag ausgeführt wird (damit ist der Rechenaufwand also linear in der Anzahl der Stellen). Hierzu bemerken wir, daß sich die Summe von zwei **Ziffern** $a, b \in \mathbb{Z}_B$ als

$$a + b = c + \gamma B, \qquad c \in \mathbb{Z}_B, \quad \gamma \in \{0, 1\}$$

ergibt, wobei γ als **Carry–Bit** bezeichnet wird. Normalerweise wird dieses bei der Ganzzahladdition ermittelt und im Prozessor gespeichert[17]. Um also zwei Multiprecision–Zahlen p, q der Länge N zu addieren, verwenden wir den folgenden (billigen) Algorithmus.

Algorithmus 2.3 (Addition) Gegeben: $p = \sum p_j B^j$, $q = \sum q_j B^j \in \mathbb{M}_w$.

[14] Die Länge einer Zahl ist also die Anzahl der **Ziffern** $\in \mathbb{Z}_B$ minus 1.

[15] "Hauptsatz der Informatik": $2^{10} = 1024 \sim 1000 = 10^3$.

[16] Und damit auch der Multiplikation.

[17] Es gibt sogar eigene Maschinensprachenbefehle, um dieses Bit zu setzen und zu löschen. Eigentlich ist die Addition eine Operation der Form $(a, b, \gamma) \to (c, \gamma)$, das heißt, man läßt das Carry–Bit einfach für die nächste Operation stehen.

1. *Setze* $\gamma = 0$.
2. *Für* $j = 0, \ldots, N$ *berechne*
$$r_j + \gamma B \leftarrow p_j + q_j + \gamma,$$
3. *Ist* $\gamma = 1$, *dann setze* $r_{N+1} = 1$.

Ergebnis: $p \oplus q =: r = \sum r_j B^j$.

Bemerkung 2.4 *1. Man sieht natürlich sofort, daß* $\ell(r) \in \{N, N+1\}$, *je nachdem, wie das Carry–Bit am Ende aussieht.*

2. Die Subtraktion funktioniert "analog". Genauer addiert man in diesem Fall Zweierkomplemente der Ausgangszahl (das entspricht mehr oder weniger einem einfachen Umdrehen der Bits in der Zahl) und behandelt wieder die Carry–Bits angemessen. Die Details wollen wir hier aber weglassen.

3. Der Rechenaufwand ist auf jeden Fall aber $O(N)$ ***Wortoperationen****.*

Etwas spannender ist dann schon die **Multiplikation** zweier Zahlen der Länge N, bei der sich

$$pq = \left(\sum_{j=0}^{N} p_j B^j\right)\left(\sum_{k=0}^{N} q_k B^k\right) = \sum_{\ell=0}^{2N}\left(\sum_{j+k=\ell} p_j q_k\right) B^\ell =: \sum_{\ell=0}^{N} r_\ell B^\ell, \tag{2.1}$$

ergibt, wobei

$$r_\ell = \sum_{j=\max\{\ell-N,0\}}^{\min\{\ell,N\}} p_j q_{\ell-j} = \sum_{j\in\mathbb{Z}} \widehat{p}_j \widehat{q}_{\ell-j}, \qquad \ell = 0, \ldots, 2N. \tag{2.2}$$

Wenn man genau hinsieht ist (2.2) nichts als die **Faltung** der beiden Folgen $\widehat{p} := (\widehat{p}_j : j \in \mathbb{Z})$ und $\widehat{q} := (\widehat{q}_j : j \in \mathbb{Z})$, die p, q einfach mit Nullen fortsetzen. Allerdings: Die so berechneten „Ziffern" r_ℓ, $\ell = 0, \ldots, 2N$, liegen nicht mehr in $\{0, \ldots, B-1\}$, sondern können entschieden größer sein[18], also muß man sich noch um Übertrag kümmern. Damit liefe die Multiplikation nach dem Motto „Falten und Übertragen" zuerst einmal auf die Bestimmung von 2N Koeffizienten (als Multiprecision–Zahlen, denn je mehr Möglichkeiten es gibt, ℓ als $j+k$ darzustellen[19], desto größer können die Zahlen werden) hinaus, die dann, wieder in Multiprecision–Arithmetik aufaddiert werden müssen – die Multiplikation mit B^ℓ ist ja "nur" ein Schiebeprozess und damit (fast) vernachlässigbar[20]. Für ein etwas effizienteres Verfahren nehmen wir an, daß unser System eine "doppeltgenaue" Multiplikation unterstützt, also

$$a \cdot b = c + d B, \qquad c, d \in \mathbb{Z}_B$$

[18] Hier ist das Rechnen mit Polynomen, das ganz analog verläuft, entscheidend einfacher.

[19] Und das sind bekanntlich $\binom{\ell+1}{j}$.

[20] Eigentlich ist die "komplexitätstheoretische" Ansicht, man müsse nur die Rechenoperationen zählen und könnte andere Aspekte wie Speicherverwaltung, Speicherzugriff etc. vernachlässigen im Zeitalter von sehr effizienten Hochleistungsrechnern nicht mehr haltbar, siehe (McGeoch, 2001). Aber irgendwie muß man ja zu *einfachen* Vergleichsmaßstäben kommen.

berechnen kann und verwenden dann die gute alte „Schulmethode“.

Übung 2.1 Formulieren Sie die Schulmethode zur Multiplikation zweier Zahlen der Länge N in der Terminologie dieser Vorlesung und weisen Sie nach, daß der Aufwand $O(N^2)$ Wortoperationen beträgt. ◊

Es stellt sich allerdings die Frage, ob die Schulmethode wirklich die schnellste Methode ist, um Multiplikationen zu berechnen. Naiv würde man erst einmal denken, daß ja schließlich in (2.1) und (2.2) jedes Produkt $p_j q_k$ von zwei Ziffern mindestens einmal verwendet wird, daher auch berechnet werden muss und daß es somit gar nicht schneller als $O(N^2)$ geht. Das ist so überzeugend wie falsch! Dazu nehmen wir an, daß $N = 2M$ gerade ist und schreiben die beiden Zahlen als

$$p = p_0 + p_1 B^M, \qquad q = q_0 + q_1 B^M,$$

also

$$pq = \left(p_0 + p_1 B^M\right)\left(q_0 + q_1 B^M\right) = p_0 q_0 + B^M (p_0 q_1 + p_1 q_0) + B^{2M} p_1 q_1.$$

Das sind vier Multiplikationen von Zahlen halber Länge, was uns aber immer noch nichts bringt. Nun machen wir eine ganz banale Beobachtung:

$$p_0 q_0 + (p_0 q_1 + p_1 q_0) + p_1 q_1 = (p_1 + p_2)(q_1 + q_2)$$

und wir stellen fest, daß

$$pq = r_0 + r_1 B^M + r_2 B^{2M}, \qquad \begin{cases} r_0 &= p_0 q_0, \\ r_2 &= p_1 q_1, \\ r_1 &= (p_0 + p_1)(q_0 + q_1) - r_0 - r_2, \end{cases} \tag{2.3}$$

ist, und wir die „Faktoren“ in dieser Identität mit drei Multiplikationen halber Länge, einer Addition halber Länge und zwei Additionen[21] ganzer Länge bestimmen können, Shifts bzw. Multiplikationen mit Potenzen von B gibt's umsonst. Jetzt setzen wir $N = 2^n$, damit wir die Halbierungsidee iterieren können und erhalten, daß sich der Aufwand $K(N)$ bei dieser Rechnung sich folgendermaßen ergibt[22]:

$$\begin{aligned} K(N) &= K(2^n) = 3\,K(2^{n-1}) + a2^n = 3\left(3K(2^{n-2}) + a2^{n-1}\right) + a2^n \\ &= 9K(2^{n-2}) + a\left(2^n + 3\,2^{n-1}\right) = \cdots = 3^k K(2^{n-k}) + a\sum_{j=0}^{k-1} 3^j 2^{n-j} \\ &= 3^n K(1) + a\sum_{j=0}^{n-1} 3^j 2^{n-j} = 3^n K(1) + a2^n \underbrace{\sum_{j=0}^{n-1}\left(\frac{3}{2}\right)^j}_{=\frac{1}{2}\left(\left(\frac{3}{2}\right)^n - 1\right)} \\ &= 3^n K(1) + \frac{a}{2}\left(3^n - 2^n\right) = 3^n\left(K(1) + \frac{a}{2}\right) - \frac{a}{2}\,2^n \end{aligned}$$

[21] Ja, eigentlich sind es Subtraktionen, aber wir sind hier nicht in der Numerik; bei Ganzzahlrechnung gibt es zwischen Addition und Subtraktion absolut **keinen** Unterschied.

[22] Wir fassen den Aufwand *aller* beteiligten Additionen in einer Komplexität aN, $a \in \mathbb{N}$, zusammen, wobei a nicht wirklich groß ist.

Übung 2.2 Verfizieren Sie die „k–Formel“ formal via Induktion. ◊

Wir sind auch schon fast fertig! Jetzt schreiben wir nur noch $3 = 2^{\log_2 3}$ und erinnern uns, daß $n = \log_2 N$ ist, denn dann ergibt sich unter Verwendung[23] von $K(1) = 1$

$$\begin{aligned} K(N) &= 2^{\log_2 3 \log_2 N}\left(K(1) + \frac{a}{2}\right) - \frac{a}{2} 2^{\log_2 N} = N^{\log_2 3}\left(1 + \frac{a}{2}\right) - N\frac{a}{2} \\ &= O(N^{\log_2 3}), \end{aligned}$$

und das ist *deutlich* besser als $O(N^2)$. Dieser einfache Trick ist das Verfahren von **Karatsuba**, siehe (Karatsuba & Ofman, 1963), das man auch in (Gathen & Gerhard, 1999; Knuth, 1998) finden kann. Aber es geht noch besser! Das schnellste Verfahren zur Multiplikation von ganzen Zahlen, das von Schönhage und Strassen (Schönhage & Strassen, 1971) entwickelt wurde, schafft es mit

$$O(N \log N) \qquad \text{bzw.} \qquad O(N \log N \log\log N)$$

Rechenoperationen, ist dann aber auch ein klein wenig anspruchsvoller in der Herleitung und Stoff für eine Computeralgebra–Vorlesung, siehe (Gathen & Gerhard, 1999; Sauer, 2001).

Bleiben also zur Vervollständigung der Grundrechenarten die Division und die Bestimmung des Divisionsrests – letztere wird entscheidend für **modulares Rechnen** werden. Wir werden aber auch sehen, daß wir für eine effiziente Realisierung der Division schon etwas mehr (auch mathematischen) Aufwand treiben müssen.

Aber auch bei der Division denkt man zuerst wieder an die sogenannte **Schulmethode**[24], also das sukzessive abdividieren „von vorne nach hinten“. Sehen wir uns das mal an einem ganz einfachen Beispiel an.

Beispiel 2.5 *Berechnung von* $12346 : 151$ *mit* $B = 10$. *Nachdem* $123 < 151$ *müssen wir also zuerst*

$$1234 : 151 = \underbrace{8 \cdot 151}_{1208} + 26$$

und dann

$$266 : 151 = \underbrace{1 \cdot 151}_{151} + 115$$

also $1234 = 81 \cdot 151 + 115$, *oder, im “altbewährten” Schema*

	1	2	3	4	6	:	151	=	81
−	1	2	0	8					
			2	6					
			2	6	6				
		−	1	5	1				
			1	1	5				

[23] Die Multiplikation zweier Zahlen der Länge 1, also zweier Worte, ist per definitionem eine Wortoperation.

[24] Dieser Begriff ist nicht meine Erfindung, sondern wurde bereits in (Schönhage & Strassen, 1971) verwendet.

So schön und einfach das Ganze ist – dieses Verfahren hat auch einiges an Nachteilen zu bieten.

Bemerkung 2.6 (Division, Schulmethode)

1. *Das „Raten" des Divisors* a *in jedem Schritt der „Schuldivision" ist eine Kunst für sich! Man kann es zwar mal mit dem Quotienten der beiden Leitziffern von* p *und* q *versuchen, aber, wie die zweite Operation in Beispiel 2.5 zeigt, man merkt erst nach einer Multiplikation (mit Ziffer) und einer Subtraktion, ob man gut geraten hat oder nicht. Gegebenenfalls muß eben* a *nach oben oder nach unten korrigiert werden.*

2. *Der Rechenaufwand ist ganz interessant: Man muß* $\ell(p) - \ell(q)$ *Multiprecision–Ziffer–Multiplikationen ausführen und dann eine Multiprecision–Subtraktion – beides ist mit einem Aufwand von* $O(\ell(q))$ *elementaren Operationen verbunden. Der Gesamtaufwand ist dann also* $O((\ell(p) - \ell(q))\,\ell(q))$ *elementare Operationen. Damit ist die Division billiger als die naive Multiplikation, aber teuerer als Kratsuba.*

Vor allem der große Rechenaufwand, aber auch die etwas haarigen Details machen dieses Verfahren nicht zum Mittel der Wahl! Was aber sonst tun? Nun, wir erinnern uns wieder mal an die Methoden der Numerischen Mathematik und verwenden das **Newton–Verfahren**, das Standardverfahren zum Lösen nichtlinearer Gleichungen der Form $f(x) = 0$, $f \in C^1(\mathbb{R})$.

Definition 2.7 *Die* ***Newton–Iteration*** *berechnet eine Folge* x_j, $j \in \mathbb{N}_0$, *über die Regel*

$$x_{j+1} = x_j - \frac{f(x_j)}{f'(x_j)}, \qquad x_0 \in \mathbb{R}.$$

Bemerkung 2.8 *Die Geometrie hinter den Newton–Iterationen ist sehr einfach: Man verwendet die Nullstelle der Tangente an die Funktion an der Stelle* x_j, *um die Nullstelle von* f *zu „raten" und iteriert dann mit dieser „geratenen" Nullstelle weiter.*

Es ist bekannt, siehe z.B. (Sauer, 2013)[25], daß, wann immer $f \in C^2(\mathbb{R})$ eine **einfache Nullstelle** x^* besitzt, d.h. $f(x^*) = 0$, $f'(x^*) \neq 0$, es eine Umgebung U von x^* gibt, so daß die Newton–Iteration für alle Startwerte $x_0 \in U$ *quadratisch* gegen x^* konvergiert, das heißt, daß

$$|x_{j+1} - x^*| \sim |x_j - x^*|^2 .$$

In vielen Anwendungen des Newton–Verfahrens besteht das größte Problem darin, die **Startumgebung** U zu finden, denn für schlechte Startwerte kann das Verfahren grandios scheitern[26]. Betrachtet man die Funktion

$$f(x) = \frac{1}{x} - a,$$

[25] Ja, solche Sachen lernt man in einer Numerik–Vorlesung.

[26] Es kann divergieren, gegen eine falsche Lösung konvergieren oder auch fröhlich im Kreis herumiterieren.

die genau die Nullstelle $x^* = \frac{1}{a}$ besitzt, dann ist $f'(x) = -\frac{1}{x^2}$ und die Iterationsvorschrift

$$x_{j+1} = x_j - \frac{x_j^{-1} - a}{-x_j^{-2}} = x_j + x_j - ax_j^2 = 2x_j - ax_j^2 \tag{2.4}$$

kommt lediglich mit Additionen[27] und Multiplikationen aus, also mit Operationen, die wir bereits kennen und effizient implementieren können. Bevor wir unseren schnellen Divisionsalgorithmus aufstellen, brauchen wir zuerst einmal ein paar Eigenschaften der Iteration (2.4).

Proposition 2.9 *Sei* $a > 0$.

1. *Es ist*
$$\left|x_{j+1} - \frac{1}{a}\right| = a\left|x_j - \frac{1}{a}\right|^2, \qquad j \in \mathbb{N}_0. \tag{2.5}$$

2. *Ist* $\rho \in (-1, 1)$ *so, daß* $x_0 = \frac{1+\rho}{a}$, *dann ist*
$$\left|x_j - \frac{1}{a}\right| \le \frac{|\rho|^{2^j}}{a}, \qquad j \in \mathbb{N}_0.$$

3. *Für die Startwerte aus* $1 + (-1, 1)/a$. *ist*
$$\left|x_j - \frac{1}{a}\right| \le \frac{2^{-2^j}}{a}, \qquad j \in \mathbb{N}_0, \tag{2.6}$$

   das Newton–Verfahren konvergiert also für diese Startwerte.

Beweis: Für 1. bemerken wir, daß

$$\begin{aligned}\left(x_j - \frac{1}{a}\right)^2 &= \frac{(ax_j - 1)^2}{a^2} = \frac{1}{a^2}\left((ax_j)^2 - 2ax_j + 1\right) = \frac{1}{a}\left(\frac{1}{a} - \left(2x_j - ax_j^2\right)\right)\\ &= \frac{1}{a}\left(\frac{1}{a} - x_{j+1}\right).\end{aligned}$$

Der Beweis von 2. ergibt sich aus

$$\left|x_0 - \frac{1}{a}\right| = \left|\frac{1 + \rho - 1}{a}\right| = \frac{|\rho|}{a} \le \frac{1}{a}$$

und, unter Verwendung von 1. und 2. für j,

$$\left|x_{j+1} - \frac{1}{a}\right| \le a\left|x_j - \frac{1}{a}\right|^2 \le a\left(\frac{\rho^{2^j}}{a}\right)^2 = \frac{1}{a}\left(\rho^{2^j}\right)^2 = \frac{\rho^{2^{j+1}}}{a}$$

[27] OK, eigentlich ist es natürlich eine Subtraktion, aber darauf soll's und nicht ankommen.

was, per Induktion, den Beweis vervollständigt.
Die Konvergenzrate (2.6) aus 3. ergibt sich schließlich daraus, daß für $\frac{1}{2a} < x_0 < \frac{3}{2a}$ die Größe ρ aus 2. die Beziehung $|\rho| < \frac{1}{2}$ erfüllt. □

Sei also nun $q \in \mathbb{M}_w$ gegeben und nehmen wir der Einfachheit an, daß q normalisiert und positiv[28] ist, daß also

$$q = \sum_{j=0}^{\ell(q)} q_j B^j \geq \underbrace{q_{\ell(q)}}_{\geq 1} B^{\ell(q)} \geq B^{\ell(q)}$$

ist. Und natürlich ist $q < B^{\ell(q)+1}$.

Um das Inverse $\frac{1}{q}$ über das Newton–Verfahren annähern zu können, wählen wir $a \in \{2, \ldots, B\}$ so, daß[29]

$$(a-1)\, q < B^{\ell(q)+1} < aq; \tag{2.7}$$

daß $a > 1$ sein muß folgt sofort aus der Tatsache, daß $q < B^{\ell(q)+1}$ ist. Wäre nun zufällig $aq = B^{\ell(q)}$, dann ist ja $\frac{1}{q} = a\, B^{-\ell(q)}$ und wir können uns das Iterieren sparen[30]. Dieses Auffinden von a entspricht übrigens gerade *einer* Ausführung des "Divisorenratens" in der Schuldivision. Damit die Notation nicht ganz so ausufert setzen wir $N := \ell(q) + 1$ und folgern aus (2.7), daß

$$\left(a\, B^{-N} - B^{-N}\right) q = \left((a-1)\, B^{-N}\right) q < 1 = \frac{1}{q}\, q < \left(a\, B^{-N}\right) q,$$

also, nach Division durch q,

$$a\, B^{-N} - B^{-N} < \frac{1}{q} < a\, B^{-N}$$

und daher

$$\frac{1}{q} < a\, B^{-N} < \frac{1}{q} + B^{-N} \tag{2.8}$$

Dies ist bereits der Schlüssel zur Wahl des Startwerts: Ist nämlich $q < \frac{1}{2} B^N$, das heißt $B^{-N} < \frac{1}{2q}$, dann setzen wir[31] $x_0 = a\, B^{-\ell(q)-1}$ und erhalten aus (2.8), daß

$$\frac{1}{q} < \underbrace{a\, B^{-\ell(q)-1}}_{=x_0} < \frac{1}{q} + \frac{1}{2q} = \frac{3}{2q}. \tag{2.9}$$

Ist hingegen $q > \frac{1}{2} B^N$, also $B^{-N} > \frac{1}{2q}$, dann ist bereits $2q > B^N$ und somit $a = 2$. Wählen wir jetzt $x = B^{-\ell(q)-1}$, dann ist

$$\frac{1}{q} - \frac{1}{2q} = \frac{1}{2q} < x_0 < \frac{1}{q}, \tag{2.10}$$

[28] Vorzeichen spielen ja beim Dividieren eine eher untergeordnete Rolle.

[29] Das heißt nichts anderes als daß wir das *kleinste* a wählen, das die rechte Ungleichung erfüllt.

[30] Dasselbe Argument greift natürlich auch im Fall $aq = B^{\ell(q)+1}$.

[31] Unter Verwendung der „alten" Notation.

was uns zusammen mit (2.9) die Abschätzung $\left|x_0 - \frac{1}{q}\right| < \frac{1}{2q}$ und damit die schnelle Konvergenz des Newton–Verfahrens liefert. Ach ja, und ist $q = \frac{B^{\ell(q)+1}}{2}$, dann ist $\frac{1}{q} = 2 \cdot B^{-\ell(q)-1}$ und das Invertieren ist wieder geschenkt.

Bemerkung 2.10 *Die Wahl*

$$x_0 = \begin{cases} a\, B^{-\ell(q)-1}, & 2q < B^{\ell(q)+1}, \\ B^{-\ell(q)-1}, & 2q > B^{\ell(q)+1}, \end{cases}$$

ist in der Praxis sehr einfach zu treffen: Man muß sich nämlich nur das höchste Bit der höchstwertigen Ziffer[32] *von* q *ansehen. Mit anderen Worten: In Wirklichkeit müssen wir sogar nur die Entscheidung*

$$x_0 = \begin{cases} a\, B^{-\ell(q)}, & q_{\ell(q)} < 2^{w-1}, \\ B^{-\ell(q)}, & q_{\ell(q)} \geq 2^{w-1}, \end{cases} \tag{2.11}$$

treffen.

Nachdem wir also nun unseren Startwert gefunden haben, rechnen wir nun die Iteration

$$x_{j+1} = -x_j \otimes (q \otimes x_j \ominus 2) \tag{2.12}$$

mit beliebig genauen, normalisierten Fließpunktzahlen weiter, also mit Zahlen der Form

$$(r, e) = r \times B^e, \qquad r \in \mathbb{M}_w, \quad e \in \mathbb{Z}.$$

wobei wir immer annehmen wollen, daß r normalisiert ist – schließlich besteht wenig Sinn darin, überflüssige Nullen herumzuschleppen. Die Multiplikation zweier solcher Zahlen ergibt sich dann als

$$(r, e) \otimes (r', e') = (r \otimes r', e + e')$$

und die Addition (mit geeignetem Schieben) im Falle von $e \geq e'$ als[33]

$$(r, e) \oplus (r', e') = \left(B^{e-e'} r \oplus r', e'\right),$$

wobei nötigenfalls noch normalisiert werden muß.

Jetzt fehlt nur noch eine **Abbruchbedingung** für die Newton–Iteration[34] Wir werden sogar sehen, daß wir die Anzahl der notwendigen Iterationen *a priori* beschränken können, und daß diese Zahl sogar verhältnismäßig klein ist. Dazu

[32] Also das höchste Bit in der Binärdarstellung des höchstwertigen Worts.

[33] Daß die Multiplikation $B^{e-e'} r$ nicht als Langszahlmultiplikation geschrieben ist, ist volle Absicht, denn diese Operation kann als einfache Schiebeoperation realisiert werden, bei der noch nicht einmal die hinten angehängten Nullen wirklich gespeichert werden müssen.

[34] Das ist bei allen iterativen Verfahren wichtig, man muss auch aufhören können, wenn es am schönsten ist. Oftmals ist die Wahl einer geeigneten Abbruchbedingung übrigens weder offensichtlich noch einfach.

erinnern wir uns zuerst an (2.6)[35] und erhalten, unter unserer "kanonischen" Voraussetzung $0 < q < p$, daß

$$|p - pqx_j| = pq\left|\frac{1}{q} - x_j\right| \le pq\frac{2^{-2^j}}{q} \le \frac{p}{2^{2^j}} \le \frac{B^{\ell(p)+1}}{2^{2^j}} = \frac{2^{w\,(\ell(p)+1)}}{2^{2^j}} = 2^{w\,(\ell(p)+1)-2^j}, \tag{2.13}$$

was für $2^j > w\ (\ell(p)+1)$, also spätestens für

$$j = 1 + \left\lfloor \log_2 w + \log_2\left(\ell(p)+1\right)\right\rfloor \tag{2.14}$$

kleiner als 1 wird. Für diesen Wert von j bezeichne $s = \left\lfloor px_j + \frac{1}{2}\right\rfloor \in \mathbb{N}$ die ganzzahlige Rundung von px_j – offensichtlich ist $|s - px_j| \le \frac{1}{2}$. Dann ist

$$|p - s\cdot q| \le |p - pqx_j| + q\underbrace{|px_j - s|}_{<\frac{1}{2}} < 1 + \frac{q}{2} \le q$$

und somit hat $r = p - s\cdot q$ die Eigenschaft, daß $|r| < q$. Ist $r < 0$, so ersetzen wir s durch $s-1$ und r durch $q + r$, was dann zwischen 0 und $q-1$ liegt, und erhalten die Darstellung

$$p = sq + r, \qquad s > 0, \quad 0 \le r < q, \tag{2.15}$$

die genau die bekannte **Division mit Rest** ist, die uns gleich noch weiter beschäftigen wird.

Bemerkung 2.11 (Division mit dem Newton–Verfahren)

1. *Zur Berechnung des Näherungsquotienten müssen nach (2.14) maximal*

$$1 + \left\lfloor \log_2 w + \log_2\left(\ell(p)+1\right)\right\rfloor$$

Iterationen durchgeführt werden, eine Zahl die vor *Beginn des Newton–Verfahrens bekannt ist. So wird aus einem iterativen Verfahren ein Algorithmus*[36] ...

2. *Da $\ell(p) < 2^{w-1}$ ist, beschränkt sich der maximale Aufwand somit auf höchstens $w+\log_2 w$ Multiprecision–Operationen. Und das ist nun wirklich nicht übermäßig viel und hängt interessanterweise nur von der Wortlänge ab. Damit ist, zumindest* komplexitätstheoretisch *die Division jetzt genauso teuer wie die Multiplikation, nur der Faktor ist natürlich signifikant größer.*

3. *Die Iteration (2.12) verlangt noch nach einer genaueren Betrachtung, denn die Stellenzahl von x_j kann ja prinzipiell so groß werden, daß man in den Multiplikationen eine Aufwand betreibt, der schlichtweg inakzeptabel wird. Und in der Tat wächst die Stellenzahl auch ganz schön flott: Sei $s_j = \ell(x_j)$ die Länge der*

[35] Achtung: Das a von dort ist jetzt unser q!

[36] Also ein Verfahren, das nach einer bestimmbaren *endlichen* Anzahl von Schritten das gewünscht Ergebnis liefert.

Mantisse von x_j, dann hat $a \otimes x_j$ ja höchsten $s_j + 1$ Stellen und $x_j \otimes (a \otimes x_j)$ maximal $2s_j + 1$ Stellen. Somit ist $s_{j+1} \leq 2s_j + 1$ und damit[37]

$$s_j \leq \left(2^{j+1} - 1\right) s_0, \qquad j \in \mathbb{N}_0,$$

Und da unser Startwert x_0 "einziffrig" gewählt war und deswegen $s_0 = 1$ ist, haben wir nach den $1 + \log_2(w\,\ell(p) + w)$ Iterationsschritten also insgesamt gerade einmal höchstens

$$2^{2+\log_2(w\,\ell(p)+w)} - 1 \leq 4w\ (\ell(p) + 1)$$

Stellen angehäuft, was immer noch ein $O(\ell(p))$ ist – es kann also nichts schlimmes passieren, insbesondere wenn man bedenkt, daß man sowieso mindestens $\ell(p)$ Stellen im "finalen" x_j benötigt, wenn $p\,x_j$ auch nur ein halbwegs vernünftiges Ergebnis sein soll.

4. *Nochmals: Das Ganze funktioniert nur deswegen so gut und mit moderatem Aufwand, weil das Newton–Verfahren* quadratisch *konvergiert und diese quadratische Konvergenz hier voll ausgenützt wird.*

5. *Die Effizienz der Newton–Iteration lebt nun im wesentlichen von der Effizienz der Multiplikation – je schneller wir also multiplizieren können, desto schneller können wir dann auch dividieren.*

Übung 2.3 Formulieren Sie den Algorithmus zur "schnellen" Division über das Newton–Verfahren im Detail.

2.2 Euklidische Ringe und modulare Arithmetik

Modulare Arithmetik ist der wesentliche Hintergrund vieler Verschlüsselungsverfahren und da schadet es nicht, wenn man die Sache ein klein wenig allgemeiner macht.

Definition 2.12 (Ring)

1. *Eine Menge R mit Operationen*[38] *„+" und „·" heißt **Ring**, wenn sie*

   (a) *eine **abelsche Gruppe** bezüglich der Addition ist: $a + b = b + a \in R$, $a, b \in R$, und es gibt ein **neutrales Element** 0 sowie zu jedem a ein $b =: -a$ mit $a + b = 0$.*

   (b) *ein **Halbgruppe** bezüglich der Multiplikation ist, d.h. $a \cdot b \in R$,*

   *und wenn die beiden Operationen über ein **Distributivgesetz**[39] miteinander verknüpft sind:*

$$a \cdot (b + c) = ab + ac, \qquad (b + c) \cdot a = ba + ca.$$

[37] Das ist eine ganz einfache Induktion.

[38] Diese Symbole stehen natürlich für Addition und Multiplikation.

[39] Das ist es, was Mathematik ausmacht: Einzelne Eigenschaften existieren nicht isoliert, sondern interagieren miteinander.

2. *Ein Ring* R *heißt* ***kommutativer Ring****, wenn die Multiplikation ebenfalls kommutativ ist,*

3. *und* ***Ring mit Einselement****, wenn es ein Element* $1 \in R$ *gibt, das bezüglich der Multiplikation neutral ist:* $a1 = 1a = a$, $a \in R$.

Beispiel 2.13 *Der klassische kommutative Ring ist* $\mathbb{Z}$*, aber natürlich auch die* ***Körper*** $\mathbb{Q}$ *und* $\mathbb{R}$*, als Beispiel für einen nichtkommutativen Ring kann man sich* $\mathbb{R}^{n\times n}$*, also die quadratischen Matrizen merken. Beides sind Ringe mit Einselement, einmal die* 1 *und einmal die Einheitsmatrizen.*

Definition 2.14 (Noch mehr Ringe)

1. *Ein Element* $a \in R$ *eines Rings* R *ist* ***Teiler*** *von* $b \in R$*, in Zeichen* $a \mid b$*, wenn es ein* $c \in R$ *gibt, so daß* $ac = b$.

2. *Ein kommutativer Ring* R *mit Einselement heißt* ***Integritätsbereich***[40] *oder* ***Integritätsring****, wenn er* ***nullteilerfrei*** *ist, das heißt, wenn es kein* $0 \neq a, b \in R$ *gibt, so daß* $ab = 0$.

3. *Ein Integritätsring* R *heißt* ***euklidischer Ring****, wenn es* ***Bewertungsfunktion*** *oder* ***euklidische Funktion*** $d : R \to \mathbb{N}_0 \cup \{-\infty\}$ *gibt, so daß es für alle* $p, q \in R$*,* $q \neq 0$*, einen* ***Faktor*** $s \in R$ *und einen* ***Rest*** $r \in R$ *gibt, so daß*

$$p = sq + r, \qquad d(r) < d(q).$$

Wir schreiben dann auch $s =: p/q$ *und* $r =: (p)_q$.

Bemerkung 2.15 (Euklidische Ringe)

1. *Beide Eigenschaften eines euklidischen Rings aus Definition 2.14 sind wichtig für das* ***praktische*** *Rechnen in Ringen. Die Nullteilerfreiheit sorgt dafür, daß man kürzen darf, das heißt, ist* $a \neq 0$ *und* $ab = ac$*, dann ist auch* $b = c$*, während die zweite Eigenschaft uns die Existenz einer Division mit Rest gibt, wobei der Rest* einfacher *ist als der Divisor.*

2. *Jede euklidische Funktion hat die Eigenschaft, daß* $d(0) < d(a)$ *für alle* $a \in R\backslash\{0\}$*. Gäbe es nämlich ein* $a \in R \setminus \{0\}$*, so daß* $d(a) \leq d(R)$*, dann erhalten wir mit* $p = q = a$*, daß eine euklidische Darstellung von* a *die Form*

$$a = sa + \underbrace{(1-s)a}_{=r}, \qquad s \in R,$$

haben muß, aber ganz egal wie wir s *wählen, würde jeder dieser Reste* $d((1-s)r) \geq d(a)$ *erfüllen und damit wäre der Ring nicht euklidisch.*

[40] Auf Englisch "**integral domain**".

3. *Es hat zwar nicht jede euklidische Funktion die Eigenschaft*

$$d(a \cdot b) \geq d(a), \qquad a, b \in R \setminus \{0\}, \tag{2.16}$$

*die man von den „normalen" euklidischen Funktionen für $\mathbb{Z}$ kennt und fast für selbstverständlich hält, aber für jeden Ring Integritätsring R gibt es so eine Bewertungsfunktion, und zwar die **minimale euklidische Funktion**, die als punktweises Minimum* aller *möglichen euklidischen Funktionen definiert ist, siehe (Gathen & Gerhard, 1999, Übung 3.5).*

4. *Der Wert $d(a) = -\infty$ ist nur für $a = 0$ zulässig, muß aber nicht angenommen werden.*

5. *Insbesondere gilt*

$$q|p \qquad \Leftrightarrow \qquad (p)_q = 0, \tag{2.17}$$

jeder euklidische Ring enthält also ganz natürlich Teilbarkeitsfragen und diese sind mit dem Rest verknüpft.

Übung 2.4 Beweisen Sie (2.17) (Einfach!). ◊

Beispiel 2.16 (Euklidische Ringe)

1. *Die ganzen Zahlen $\mathbb{Z}$ bilden zusammen mit der Funktion $d = |\cdot|$ einen euklidischen Ring.*

2. *Die Polynome $\mathbb{K}[x]$ über einem Körper $\mathbb{K}$ bilden einen euklidischen Ring mit der Funktion $d = \deg$, wobei $\deg 0 = -\infty$.*

3. *Ein Körper $\mathbb{K}$ ist ein euklidischer Ring mit $d = (1 - \delta_0)$.*

4. *Eine etwas obskurere euklidische Funktion auf $\mathbb{Z}$ ist $d(3) = 2$ und $d = |\cdot|$ sonst. Diese euklidische Funktion*[41] *erfüllt* nicht *die Bedingung (2.16), da $d(-1 \cdot 3) = d(-3) = 3 > 2 = d(3)$ ist. Trotzdem ist aber immer noch $d(0)$ minimal ...*

In jedem beliebigen Ring[42] können wir zwei wichtige Größen definieren, nämlich ein **größter gemeinsamer Teiler** $\mathrm{ggT}(a, b)$ und ein **kleinstes gemeinsames Vielfaches** $\mathrm{kgV}(a, b)$ von zwei Elementen $a, b \in R$. Das geschieht durch die Forderungen

$$\begin{array}{lll} \mathrm{ggT}(a,b) \,|\, a, & \mathrm{ggT}(a,b) \,|\, b, & c \,|\, a,\ c \,|\, b \Rightarrow c \,|\, \mathrm{ggT}(a,b), \\ a \,|\, \mathrm{kgV}(a,b), & b \,|\, \mathrm{kgV}(a,b)b, & a \,|\, c,\ b \,|\, c \Rightarrow \mathrm{kgV}(a,b) \,|\, c. \end{array} \tag{2.18}$$

Dabei sind jeweils die ersten beiden Eigenschaften für die Eigenschaft „gemeinsamer Teiler" bzw. „gemeinsames Vielfaches" zuständig, während die dritte Eigenschaft für das „größter" bzw. das „kleinstes" sorgt. Nachdem nicht so

[41] Euklidisch ist sie dadurch, daß der Divisionsrest bei Division in $\{-1, 0, 1\}$ gewählt wird.
[42] Hier brauchen wir also keinen euklidischen Ring.

ganz klar ist, ob das Nullelement eines Rings ein anderes Element teilt oder nicht, *definieren* wir außerdem, daß

$$\mathrm{ggT}(a,b) = 0 \qquad \text{falls} \qquad a = 0 \text{ oder } b = 0, \tag{2.19}$$

was natürlich auch den Fall $a = b = 0$ einschließen soll[43].

Wie gesagt, definieren können wir ggT und kgV auf beliebigen Ringen, zum Ausrechnen ist jedoch ein euklidischer Ring hilfreich.

Algorithmus 2.17 (Euklidischer Algorithmus)
Gegeben: $a, b \in R$, R *euklidischer Ring.*
Solange $b \neq 0$ *setze*

$$c \leftarrow (a)_b, \qquad a \leftarrow b, \qquad b \leftarrow c.$$

Ergebnis: $a = \mathrm{ggT}(a,b)$.

Proposition 2.18 *Der euklidische Algorithmus terminiert für alle* $a, b \in R$ *nach endlich vielen Schritten und berechnet* $\mathrm{ggT}(a,b)$.

Beweis: Wir schreiben den Algorithmus als

$$r_{j+1} = (r_{j-1})_{r_j}, \qquad j \in \mathbb{N}, \qquad r_0 = a,\ r_1 = b.$$

Da $d(r_{j+1}) < d(r_j)$ bricht das Verfahren nach endlich vielen Schritten ab. Die Beziehung können wir auch schreiben als

$$r_{j+1} = r_{j-1} - s_j\, r_j, \qquad s_j \in R, \qquad j \in \mathbb{N}, \tag{2.20}$$

woraus induktiv sofort folgt, daß $\mathrm{ggT}(a,b) \mid r_j$, $j \in \mathbb{N}_0$. Ist nun $r_{n+1} = 0$ und $r_n \neq 0$, dann gilt $r_n \mid r_{n-1}$, also teilt r_n wegen

$$r_{j-1} = s_j r_j + r_{j+1}, \qquad j = n-1, \ldots, 1,$$

auch $r_{n-2}, r_{n-3}, \ldots, r_1 = b, r_0 = a$ und damit ist $r_n = \mathrm{ggT}(a,b)$. □

Bemerkung 2.19

1. *Indem wir den Divisionsrest bei Division durch* q *nicht in der Menge* $\{0, \ldots, q-1\}$ *sondern in*

$$\left\{-\frac{q}{2}+1, \ldots, \frac{q}{2}\right\}, \qquad q \in 2\mathbb{Z} \setminus \{0\},$$

$$\left\{-\frac{q-1}{2}, \ldots, \frac{q-1}{2}\right\}, \qquad q \in 2\mathbb{Z}+1,$$

wählen, erhalten wir einen schnelleren Abfall der Bewertungsfunktion als im normalen euklidischen Algorithmus. Außerdem sehen wir dabei, daß sich die Bewertungsfunktion in jedem Iterationsschritt nicht um 1 *verringert, wie im Beweis verwendet, sondern halbiert, so daß der euklidische Algorithmus hier sogar in* $\log_2 q$ *Schritten* garantiert.

[43] Hier sind wir ebenfalls wieder konsistent mit der anschaulichen Tatsache, daß $\mathrm{ggT}(a,a) = a$ ist.

2. *Diese schnelle Konvergenz liegt aber auch vor, wenn man die Standardbewertung mit nichtnegativem Rest verwendet. Ist nämlich* $p = qs + r$*, dann ist für* $q \leq \frac{p}{2}$ *trivialerweise auch* $r < q \leq \frac{p}{2}$*, ist hingegen* $q > \frac{p}{2}$*, dann muss* $s = 1$ *sein und somit ist* $r = p - q < \frac{p}{2}$*. Also gilt immer* $r < \frac{p}{2}$ *und wir haben wieder Terminierung nach einer logarithmischen Anzahl von Schritten.*

Beispiel 2.20 *Betrachten wir die Berechnung von* $\mathrm{ggT}(21, 13) = 1$*. Mit dem "naiven" Verfahren erhalten wir die Folge*

j	r_j	r_{j+1}
0	21	13
1	13	8
2	8	5
3	5	3
4	3	2
5	2	1

mit vorzeichenbehaftetem Divisionsrest hingegen ergibt sich

j	r_j	r_{j+1}
0	21	13
1	13	-5
2	-5	-2
3	-2	1

was immer noch etwas schneller ist.

Bemerkung 2.21 *Besonders viele Iterationsschritte benötigt der euklidische Algorithmus für die* Fibonaccizahlen $x_{n+1} = x_n + x_{n-1}$, $x_0 = x_1 = 1$, *siehe (Knuth, 1998).*

Richtig schön wird der euklidische Algorithmus aber erst dann, wenn man ihn noch ein bisschen aufbohrt.

Algorithmus 2.22 (Erweiterter Euklidischer Algorithmus)
Gegeben: $a, b \in R$, R *euklidischer Ring mit Normalform.*

1. *Initialisiere:*

$$\begin{array}{lll} r_0 \leftarrow a, & p_0 \leftarrow 1, & q_0 \leftarrow 0, \\ r_1 \leftarrow b, & p_1 \leftarrow 0, & q_1 \leftarrow 1. \end{array} \tag{2.21}$$

   und $j \leftarrow 1$.

2. *Solange* $r_j \neq 0$

   (a) *Division mit Rest:*

$$s \leftarrow r_{j-1} \,/\, r_j, \qquad r_{j+1} \leftarrow (r_{j-1})_{r_j}\,.$$

(b) Update von p *und* q

$$p_{j+1} \leftarrow (p_{j-1} - s\, p_j)\,, \qquad q_{j+1} \leftarrow (q_{j-1} - s\, q_j)\,, \tag{2.22}$$

(c) Setze $j \leftarrow j + 1$.

3. *Setze* $j \leftarrow j - 1$.

Ergebnis: $r_j = \mathrm{ggT}\,(a, b)$ *und* $(p, q) = (p_j, q_j) \in R^2$*, so daß*

$$p\, a + q\, b = \mathrm{ggT}\,(a, b)\,. \tag{2.23}$$

Bemerkung 2.23 (Erweiterter Euklid)

1. *Die* ***Bézout–Koeffizienten*** p, q *sind ein mehr als schönes Abfallprodukt des erweiterten euklidischen Algorithmus. Sind nämlich* a, b *teilerfremd, also* $\mathrm{ggT}\,(a, b) = 1$*, dann ist* $q\, b \equiv 1$ *modulo* a *und damit ist* $q \equiv b^{-1}$ *modulo* a*. So ganz nebenbei haben wir also auch das Problem gelöst, wie wir in dem endlichen Körper* $\mathbb{F}_p = \mathbb{Z}/\langle p\rangle$*,* $p \in \mathbb{N}$ *prim, invertieren.*

2. *Man könnte die Bézout–Koeffizienten auch mit dem „klassischen" Verfahren bestimmen, den euklidischen Algorithmus „von hinten aufzurollen", indem man die Zerlegungen* $r_{j+1} = r_{j-1} - s_j r_j$ *wieder einsetzt:*

$$\begin{aligned} \mathrm{ggT}\,(a, b) &= r_n = r_{n-2} - s_{n-1} r_{n-1} = r_{n-2} - s_{n-1}\,(r_{n-3} - s_{n-2} r_{n-2}) \\ &= -s_{n-1} r_{n-3} + (1 + s_{n-1} s_{n-2})\; r_{n-2} \\ &= p_j\, r_{j-1} + q_j\, r_j, \qquad j = n-1, \ldots, 1, \end{aligned}$$

   wobei man auf die Rekursionsformeln

$$p_{j-1} = q_j \qquad \textit{und} \quad q_{j-1} = p_j - s_{j-1} q_j$$

   zurückgreifen kann. Um diese Rückwärtsrekursion aber rechnen zu können, müßte man alle Werte r_j*,* s_j*, die man im Laufe des euklidischen Algorithmus erhalten hat, zwischenspeichern, was sicherlich ineffizient ist.*

3. *Was in Bemerkung 2.19 gesagt wurde, gilt sinngemäß natürlich auch für den erweiterten euklidischen Algorithmus.*

Satz 2.24 *Sei* R *ein euklidischer Ring mit Normalform und* $a, b \in R$*. Der erweiterte euklidische Algorithmus terminiert nach einer endlichen Anzahl von Schritten und berechnet* $\mathrm{ggT}\,(a, b)$*, und Zahlen* p, q*, so daß*

$$p\, a + q\, b = \mathrm{ggT}\,(a, b)\,. \tag{2.24}$$

ist.

Beweis: Konvergenz und ggT folgen direkt aus Proposition 2.18, (2.24) ist eine Konsequenz aus der allgemeineren Invariante

$$p_j\, a + q_j\, b = r_j, \qquad j \in \mathbb{N}_0, \tag{2.25}$$

die wir per Induktion beweisen wollen. Für $j = 0, 1$ ist (2.25) gemäß der Initialisierung (2.21) erfüllt:

$$r_0 = a = a + 0\, b, \qquad \text{und} \qquad r_1 = b = 0\, a + b.$$

Ansonsten verwenden wir (2.22) für

$$\begin{aligned} p_{j+1}\, a + q_{j+1}\, b &= (p_{j-1} - s\, p_j)\; a + (q_{j-1} - s\, q_j)\; b \\ &= p_{j-1} a + q_{j-1} b - s_j\, (p_j a + q_j b) = r_{j-1} - s_j r_j = (r_{j-1})_{r_j} \\ &= r_{j+1}, \end{aligned}$$

und das war's dann auch schon. □

Beispiel 2.25 *Jetzt aber erst einmal ein Beispiel für den erweiterten euklidischen Algorithmus.Sei* $R = \mathbb{Z}$, $a = 126$, $b = 35$. *Wir betrachten das folgende Tableau*

# *It.*	r_{j-1}	r_j	p_j	q_j
0	0	126	1	0
1	126	35	0	1

Nun berechnen wir, daß $126 = 3 \times 35 + 21$, *also ist* $r_2 = 35$, *sowie*

$$p_2 = 1 - 3 \times 0 = 1 \qquad \textit{und} \qquad q_2 = 0 - 1 \times 3 = -3,$$

was unser Tableau zu

# *It.*	r_{j-1}	r_j	p_j	q_j
0	0	126	1	0
1	126	35	0	1
2	35	21	1	−3

erweitert. Und in der Tat: $126 - 3 \times 35 = 21$. *Ziehen wir das so weiter durch, so erhalten wir*

# *It.*	r_{j-1}	r_j	p_j	q_j
0	0	126	1	0
1	126	35	0	1
2	35	21	1	−3
3	21	14	−1	4
4	14	7	2	−7

was uns auch die Darstellung $\mathrm{ggT}\,(126, 35) = 7 = 2 \times 126 - 7 \times 35$ *liefert.*

Übung 2.5 Implementieren Sie den erweiterten euklidischen Algorithmus in `Matlab/Octave`. ◊

2.3 Modulares Rechnen

Nach unseren Vorarbeiten haben wir jetzt also die Algorithmen beisammen, um in Restklassenringen rechnen zu können. Dazu fassen wir erst noch einmal zusammen.

Definition 2.26 *Sei* R *ein Ring und* $m \in R$.

1. $a \in R$ *heißt* ***Einheit****, wenn* $a^{-1} \in R$ *existiert. Mit* $R^\times$ *bezeichnen wir die Menge aller Einheiten in* R.
2. *Wir sagen, zwei Elemente* $a, b \in R$ *sind* ***kongruent modulo*** m*, in Zeichen* $a \equiv_m b$*, wenn* $a - b \in \langle m \rangle := mR$.
3. *Für* $a \in R$ *ist die* ***Restklasse*** *oder* ***Äquivalenzklasse*** *zu* a *modulo* m *die Menge*
$$[a] := [a]_m := \{b \in R \ : \ a \equiv_m b\}.$$
4. *Der* ***Restklassenring*** $R/\langle m \rangle = R/mR$ *ist die Menge aller Restklassen mit den Operationen*
$$[a] \cdot [b] = [a \cdot b] \qquad \textit{und} \qquad [a] + [b] = [a + b].$$

Bemerkung 2.27 (Restklassenringe)

1. *Der "Standardfall" ist natürlich* $R = \mathbb{Z}$ *und die Verwendung des Divisionsrests. Dann ist, für* $p \in \mathbb{Z}$*, der Restklassenring*
$$\mathbb{Z}/\langle p \rangle = \mathbb{Z}_p = \{0, \ldots, p-1\}$$
und wir haben für $a, b \in \mathbb{Z}_p$ *die Rechenoperationen*
$$a + b := (a + b)_p \qquad \textit{und} \qquad a \cdot b := (a \cdot b)_p.$$
2. *Diese Vorgehensweise können wir auf jeden euklidischen Ring übertragen, bei dem der Divisionsrest eindeutig geregelt ist, also insbesondere auf die Polynome!*
3. *Ist der Modulus* m *eine Einheit, also* $m^{-1} \in R$*, dann gibt es genau eine Restklasse, denn in diesem Fall ist* $mR = mm^{-1}R = 1R = R$ *und je zwei beliebige Elemente des Ringes sind kongruent modulo* m.
4. *Generell sagen uns die Einheiten durchaus etwas über die Struktur eines Rings. Bei einem Körper* $\mathbb{K}$ *haben wir die maximale Situation* $\mathbb{K}^\times = \mathbb{K} \setminus \{0\}$*, bei* $\mathbb{Z}$ *auf der anderen Seite die minimale Situation* $\mathbb{Z}^\times = \{\pm 1\}$.

Übung 2.6 Zeigen Sie: Ist R ein kommutativer Ring mit Einselement, dann gilt immer
$$\{\pm 1\} \subseteq R^\times \subseteq R \setminus \{0\}.$$
◊

Übung 2.7 Gibt es einen Ring R mit $\#R^\times = 1$? ◊

Eine schöne Eigenschaft von Restklassenringen besteht darin, daß wir ihre Einheiten sehr einfach angeben können und daß sich **algorithmisch** entscheiden läßt, ob ein Element eine Einheit ist.

Proposition 2.28 *Sei* R *ein euklidischer Ring und* $m \in R$. *Dann ist die Äquivalenzklasse* $[a]$ *genau dann eine Einheit in* $R/\langle m\rangle$, *wenn* $\mathrm{ggT}(a, m) = 1$.

Beweis: Ist $\mathrm{ggT}(a, m) =: b \neq 1$, dann gibt es $p, q \in R$, $[q] \neq [0]$, so daß $a = bp$, $m = bq$, und damit ist $[q] \cdot [a] = [qa] = [bpq] = [pm] = [0]$, weswegen $[a]$ keine Einheit sein kann. Ist andererseits $\mathrm{ggT}(a, m) = 1$, dann verwenden wir unseren erweiterten euklidischen Algorithmus 2.22 und erhalten $p, q \in R$, so daß

$$1 = \mathrm{ggT}(a, m) = pa + qm \equiv_m p\, a,$$

so daß $a^{-1} = p$ in $R/\langle m\rangle$. □

Korollar 2.29

$$\mathbb{Z}_m^\times = \{a \in \mathbb{Z}_m \ : \ \mathrm{ggT}(a, m) = 1\}.$$

Wenn also die Anzahl der Einheiten in einem Ring so wichtig ist, dann ist es kein Wunder, wenn sowas auch einen besonderen Namen hat, zumindest für besondere Ringe.

Definition 2.30 *Die* ***Euler–Funktion*** $\varphi : \mathbb{N} \to \mathbb{N}$, *ist definiert als*

$$\varphi(m) := \#\mathbb{Z}_m^\times = \#\{a \in \mathbb{Z}_m \ : \ \mathrm{ggT}(a, m) = 1\}. \tag{2.26}$$

Besonders schön sind natürlich Ringe, in denen man nach Herzenslust auch dividieren darf, also Ringe, in denen alle von Null verschiedenen Elemente Einheiten sind.

Definition 2.31 *Sei* R *ein euklidischer Ring.*

1. *Ein Element* $a \in R$ *heißt* ***irreduzibel****, wenn*

$$\mathrm{ggT}(a, b) = 1, \qquad b \in R \setminus \{0, a\}.$$

2. *Man bezeichnet den Ring* R *als* ***Körper****, wenn* $R^\times = R \setminus \{0\}$.

Korollar 2.32 *Sei* R *ein euklidischer Ring mit Normalform. Für* $m \in R$ *ist* $R/\langle m\rangle$ *genau dann ein Körper, wenn* m *irreduzibel ist.*
Insbesondere ist $\mathbb{F}_p := \mathbb{Z}_p = \mathbb{Z}/\langle p\rangle$ *ein Körper wenn* p *eine Primzahl ist.*

Übung 2.8 Zeigen Sie: Ist m reduzibel, dann hat $R/\langle m\rangle$ einen Nullteiler.

Beispiel 2.33 (Hashing) *Eine ganz einfache Anwendung modularen Rechnens ist das* ***Hashing*** *oder* ***Fingerprinting****: Man sortiert große Datensätze (z.B. digitalisierte Fingerabdrücke, DNS–Daten) nicht systematisch, sondern speichert zusätzlich die Information "modulo* p*" für eine oder mehrere Primzahlen. Solche Datensätze kann man als riesige Zahlen* $a, b \in \mathbb{N}$ *interpretieren und anstatt sofort nachzuprüfen, ob* $a = b$ *gilt, testet man zuerst einmal, ob*

$$a \equiv_{p_1} b, \quad a \equiv_{p_2} b, \quad \ldots \quad a \equiv_{p_n} b$$

für sinnigerweise unterschiedliche (Prim-) Zahlen $p_1, \ldots, p_n$ *erfüllt ist. Diese Test kann man parallelisieren und die Hashwerte* $(a)_{p_j}$, $j = 1, \ldots, n$, *vorberechnen. Geht dann auch nur einer der Tests schief*[44]*, dann kann man sich den großen Vergleich sparen.*

[44] Was hier sozusagen der Erfolgsfall ist.

Endliche Körper sind natürlich eine feine Sache, denn sie sind einfach auf dem Rechner darzustellen. Daß es zu jeder Primzahl p einen endlichen Körper $\mathbb{F}_p$ gibt, wissen wir bereits, aber das geht noch ein bisschen allgemeiner.

Proposition 2.34 *Sei $p \in \mathbb{N}$ eine Primzahl und $f \in \mathbb{F}_p[x]$ ein irreduzibles Polynom vom Grad n. Dann ist $\mathbb{F}_p[x]/\langle f\rangle$ ein endlicher Körper mit p^n Elementen.*

Beweis: Daß $\mathbb{F}_p[x]/\langle f\rangle$ ein Körper ist ist, folgt sofort mit einer zweimaligen Anwendung von Korollar 2.32. Da außerdem für $f(x) = x^n + \cdots \in \Pi_n$ die Beziehung[45] $\mathbb{K}/\langle f\rangle \simeq \Pi_{n-1} \simeq \mathbb{K}^n$ gilt, ist also auch $\mathbb{F}_p/\langle f\rangle \simeq \mathbb{F}_p^n$ und daher

$$\#\mathbb{F}_p/\langle f\rangle = (\#\mathbb{F}_p)^n = p^n.$$

□

2.4 Exkurs: Geteilte Geheimnisse und Interpolation

Eine erste Anwendung des Gelernten in einem kryptographischen Kontext und gleichzeitig ein sehr netter Bezug zur numerischen Mathematik (Sauer, 2013) ist das folgende Problem:

> *Eine geheime Information, codiert in eine Zahl $g \in \mathbb{N}$, soll so in N Daten verschlüsselt ("geteilt") werden, daß das Geheimnis zwar aus allen N Informationen, nicht aber aus $N-1$ der Werte rekonstruiert werden kann.*
>
> *Allgemeiner könnte man fordern, daß zur Rekonstruktion des Geheimnisses $M \le N$ der N Informationen, nicht aber $M-1$ ausreichend sind.*

Zugegeben, ein bißchen erinnert das Ganze an die alten Edgar–Wallace–Filme wie "Die Tür mit den sieben Schlössern", aber man kann sich durchaus auch ernsthafte Anwendungen vorstellen. Wie dem auch sein, die mathematische Methode ist hier ein numerischer Klassiker, nämlich die **Polynominterpolation**. Sei $\mathbb{K}$ ein Körper, so daß $g \in \mathbb{K}$, z.B., $\mathbb{K} = \mathbb{Q}$ oder $\mathbb{K} = \mathbb{F}_p := \mathbb{Z}/\langle p\rangle$, wobei $p > g$ prim, und $\mathbb{K}[x]$ wieder der Ring der Polynome über $\mathbb{K}$.

Lemma 2.35 *Es sei $N \in \mathbb{N}$ und $\mathbb{K}$ ein Körper. Für jede Wahl von $N+1$ disjunkten Punkten[46] $x_0, \ldots, x_N \in \mathbb{K}$ und Werten $y_0, \ldots, y_n$ gibt es genau ein Polynom $f \in \mathbb{K}[x]$ vom Grad $\le N$, so daß*

$$f(x_j) = y_j, \qquad j = 0, \ldots, N.$$

Beweis: Um die Existenz zu beweisen, verwendet man normalerweise die **Lagrange–Basispolynome**

$$\ell_j(x) := \prod_{k \ne j} \frac{x - x_k}{x_j - x_k}, \qquad j = 0, \ldots, N,$$

[45] Dieser Isomorphismus hängt nur von *Grad* von f, nicht von f selbst ab, das heißt, alle diese Körper sind zueinander isomorph – aber man muß natürlich die Rechenregeln entsprechend anpassen.

[46] Das bedeutet natürlich, daß $\mathbb{K}$ mindestens $N+1$ Elemente haben muß.

die alle vom Grad $\leq N$ sind und die Eigenschaft $\ell_j(x_k) = \delta_{jk}$, $j,k = 0,\ldots,N$, haben, denn dann ist

$$f(x) = \sum_{j=0}^{N} y_j\, \ell_j(x).$$

Die Eindeutigkeit ergibt sich sofort, wenn man berücksichtigt, daß die Differenz zweier Lösungen $f - \widetilde{f}$, ein Polynom vom Grad $\leq N$ ist, das an den $N+1$ Punkten $x_0,\ldots,x_N$ verschwindet, also das Nullpolynom sein muß[47]. □

Übung 2.9

1. Zeigen Sie, daß ein Polynom $f \in \mathbb{K}[x]$ über einem Körper $\mathbb{K}$ genau dann an einem Punkt $\xi \in \mathbb{K}$ verschwindet, wenn es einen Linearfaktor $x - \xi$ enthält:
$$f(\xi) = 0 \qquad \Leftrightarrow f(x) = (x-\xi)\; g(x), \quad g \in \mathbb{K}[x].$$
Hinweis: Verwenden Sie Polynomdivision mit Rest.

2. Folgern Sie daraus, daß ein Polynom vom Grad N, das an $N+1$ disjunkten Punkten verschwindet, das Nullpolynom sein muß.

♢

Das legt auch schon die Strategie für die Ver- und Entschlüsselung des geteilten Geheimnisses nahe: Wir wählen

1. eine Primzahl $p > \max\{g, N+1\}$,
2. $N-1$ (zufällige oder beliebige) Werte $f_1,\ldots,f_{N-1} \in \mathbb{F}_p$,
3. N disjunkte Punkte $x_1,\ldots,x_N \in \mathbb{F}_p \setminus \{0\}$, einen für jeden Geheimnisträger[48].

Die letzte Bedingung ist der Grund, warum $p > N+1$ gefordert wurde, denn der Körper $\mathbb{F}_p$ muß ja mindestens die N verschiedenen Werte $x_1,\ldots,x_N$ und den Wert 0 enthalten[49]. Dann betrachtet man das Polynom

$$f(x) = g + \sum_{j=1}^{N-1} f_j\, x^j, \qquad \text{d.h.} \qquad g = f(0).$$

Die j–te Teilinformation, d.h., der j–te Schlüssel $j = 1,\ldots,N$, besteht nun in dem Wert $g_j = f(x_j)$ und die Rekonstruktion ergibt sich als

$$g = f(0) = \sum_{j=1}^{N} g_j\, \ell_j(0),$$

[47] In der Numerik wird diese Tatsache gerne über den Satz von Rolle bewiesen – der aber setzt voraus, daß wir einen Körper $\mathbb{K}$ verwenden, über dem man Analysis betreiben kann, also etwa $\mathbb{K} = \mathbb{R}$. Und stetige Funktionen setzen ein gewisses Maß an Topologie voraus, um vernünftig definiert zu werden. Trotzdem gilt die Aussage, siehe Übung 2.9

[48] Der Nullpunkt ist für das Geheimnis reserviert.

[49] Das Nullelement eines Körpers, ganz egal, welches Symbol wir dafür verwenden, ist ja immer Bestandteil der Körperdefinition und damit festgelegt.

wobei die Werte $\ell_j(0)$ für die Rekonstruktion vorberechnet werden können. Da $f(0)$ das Geheimnis darstellt, ist es nicht sinnvoll, daß keiner der Punkte x_j, $j = 0, \ldots, N$, der Nullpunkt ist, denn sonst gibt man einem "Diktator" alle Information und den anderen nur vollkommen irrelevantes Wissen.

Ist dieser Code nun sicher? Wenn wie annehmen, daß nur $g_1, \ldots, g_{N-1}$ bekannt sind und mit

$$\widehat{f}(x) := \sum_{j=1}^{N-1} g_j \, \ell_j(x) = \sum_{j=1}^{N-1} g_j \prod_{\substack{k=1 \\ k \neq j}}^{N-1} \frac{x - x_k}{x_j - x_k}$$

dasjenige Polynom bezeichnen, das die Werte $g_1, \ldots, g_{N-1}$ an den Stellen $x_1, \ldots, x_{N-1}$ interpoliert, dann hat jedes Polynom, das an $x_1, \ldots, x_{N-1}$ die Werte $g_1, \ldots, g_{N-1}$ annimmt, die Form

$$f_y(x) = \widehat{f}(x) + y \underbrace{(x - x_1) \cdots (x - x_{N-1})}_{=:\omega(x)}, \qquad y \in \mathbb{F}_p,$$

und da $\omega(0) = x_1 \cdots x_{N-1} \neq 0$, ist

$$\{f_y(0) \, : \, y \in \mathbb{F}_p\} = \mathbb{F}_p,$$

das heißt, *alle* konstanten Terme ließen sich aus der unvollständigen Information ableiten. Andererseits gibt es aber nur *genau einen* richtigen Wert, nämlich den, den man erhält wenn man verlangt, daß

$$f_y(x_N) = g_N, \qquad \text{also} \qquad y = \frac{\widehat{f}(x_N)}{\omega(x_N)}$$

ist. Also ist das Geheimnis sicher, denn die Kenntnis eines Polynoms vom Grad $N - 1$ an $N - 1$ Punkten aus $\mathbb{F}_p \setminus \{0\}$ erlaubt keinen Rückschluss auf den Wert des Polynoms an $x = 0$.

Der Fall, daß bereits $M \leq N$ Leute das Geheimnis entschlüsseln können ist nun eine einfache Folgerung: Wir nehmen eben kein Polynom vom Grad $N - 1$, sondern lediglich eines vom Grad $M - 1 \leq N - 1$.

Übung 2.10 Implementieren Sie die Verschlüsselung von gemeinsamen Geheimnissen für Parameter $M \leq N$ in `Matlab` oder `Octave`. ◊

Übung 2.11 Ist es für dieses Verfahren wichtig, daß die Stellen $x_1, \ldots, x_N$ geheimgehalten werden oder nicht? Begründen Sie Ihre Antwort. ◊

2.5 Der chinesische Restsatz

In Beispiel 2.33 haben wir ein Element eines Rings durch mehrere Moduln $(a)_{p_j}$ ersetzt und es stellt sich natürlich die Frage, wann und in welchem Rahmen man diesen Prozess umkehren kann. Wir stehen also vor dem folgenden Problem:

> *Wie kann man die Zahl* q *aus den Divisionsresten* $r_j := (q)_{q_j}$, $j = 1, \ldots, n$, *rekonstruieren?*
>
> *Genauer, wenn Primzahlpotenzen* $q_j = p_j^{e_j}$, $j = 1, \ldots, n$ *vorgegeben sind, wie findet man zu gegebenem ein* $q' \in \mathbb{N}$, *so daß* $q \equiv_m q'$ *und* $q \equiv_{q_j} q'$, $j = 1, \ldots, n$, *wobei* $m = \prod q_j$.

Die Antwort ist erstaunlich einfach und verwendet – nicht ganz zufällig – dieselbe Idee wie bei der Polynominterpolation in Lemma 2.35. Dazu konstruieren wir zuerst „Lagrange–Zahlen“ $\ell_j \in \mathbb{Z}_m$ mit der Eigenschaft, daß

$$\ell_j \equiv_{q_k} \delta_{jk}, \qquad j,k = 1,\ldots,n, \tag{2.27}$$

denn dann gilt

$$q' = \left(\sum_{j=1}^{n} r_j\,\ell_j\right)_m \qquad \Rightarrow \qquad q' = \underbrace{r_0\,m}_{\equiv_{q_j} 0} + \sum_{k=1}^{n} r_k\,\underbrace{\ell_k}_{\equiv_{q_j}\delta_{jk}} \equiv_{q_j} r_j.$$

Die Berechnung der ℓ_j ist einfach: Da die Zahlen[50] $m_j := p_j^{-e_j} m = \frac{m}{q_j}$, $j = 1,\ldots,n$, die Eigenschaft

$$\mathrm{ggT}\left(m_j, p_j^{e_j}\right) = \mathrm{ggT}\,(m_j, q_j) = \mathrm{ggT}\,(m_j, p_j) = 1, \qquad j = 1,\ldots,n, \tag{2.28}$$

haben, können wir $(m_j)_{q_j}$ in $\mathbb{Z}_{q_j}$ mit dem erweiterten euklidischen Algorithmus invertieren[51] und erhalten Zahlen $s_j \in \mathbb{Z}$, $j = 1,\ldots,n$, so daß $s_j\,m_j \equiv_{q_j} 1$, $j = 1,\ldots,n$, und damit sind

$$\ell_j := (s_j\,m_j)_m\,, \qquad j = 1,\ldots,n, \tag{2.29}$$

die gesuchten Faktoren. Damit haben wir also das folgende Rekonstruktionsverfahren.

Algorithmus 2.36 *(CRT[52]–Algorithmus für $\mathbb{Z}$)*
Gegeben: *Paarweise teilerfremde Zahlen*[53] $q_1,\ldots,q_n$ *und Werte* $r_j \in \mathbb{Z}_{q_j}$, $j = 1,\ldots,n$.

1. *Setze* $m = q_1 \cdots q_n$.
2. *Bestimme mit dem erweitertem euklidischen Algorithmus Zahlen* $a_j, b_j \in \mathbb{Z}$, *so daß*
$$a_j\,\frac{m}{q_j} + b_j\,q_j = \mathrm{ggT}\,(m/q_j, q_j) = 1,$$
und setze
$$\ell_j \leftarrow \left(a_j\,\frac{m}{q_j}\right)_m, \qquad j = 1,\ldots,n.$$

[50] Diese Zahlen sind dann die Produkte der anderen Primfaktoren, wenn man einmal einen festgehalten hat.

[51] Damit das auch wirklich immer funktioniert, mussten wir die q_j als Primzahlpotenzen wählen, siehe Proposition 2.34.

[52] *CRT* steht für “Chinese Remainder Theorem”, die englische Bezeichnung für den chinesischen Restsatz.

[53] Diese Zahlen brauchen keine Potenzen unterschiedlicher Primzahlen sein, paarweise teilerfremd oder *koprim* reicht völlig. Entscheidend ist ja nur, daß $\mathrm{ggT}\,(m_j, q_j) = 1, j = 1,\ldots,n$, denn dann kann man in $\mathbb{Z}_{q_j}$ invertieren.

3. *Setze*

$$q \leftarrow \left(\sum_{j=1}^{n} r_j \, \ell_j \right)_m$$

Ergebnis: $q \in \mathbb{Z}_m$ *so daß* $q \equiv_{q_j} r_j$, $j = 1, \ldots, n$.

Bemerkung 2.37

1. *Algorithmus 2.36 hat eine ziemliche praktische Bedeutung: Wir können eine große Zahl in* $\mathbb{Z}_m$ *in ihre Reste modulo der einfachen Primfaktoren von* m *zerlegen, die Rechnung in diesen wesentlich kleineren Körpern durchführen und brauchen schließlich "nur" das Ergebnis wieder zusammenzusetzen.*

2. *Was für praktische Anwendungen ebenfalls sehr ansprechend ist, ist die Tatsache, daß die "Zwischenrechnungen" in den individuellen Restklassenringen* $\mathbb{Z}_{q_j}$, $j = 1, \ldots, n$, *nun komplett voneinander unabhängig und damit* parallelisierbar *sind! Wählt man außerdem die Zahlen* q_j *alle so, daß* $\ell(q_j) = 0$ *(in einer Multiprecision–Arithmetik), dann habe alle Primfaktoren nur eine* B*-adische Stelle und daher besteht das ganze Restklassenrechnen nur aus* elementaren *Rechenoperationen.*

3. *Stehen die Zahl* m *und damit ihre Primfaktoren* a priori *fest, dann können die Zahlen* ℓ_j, $j = 1, \ldots, n$, *als "Invarianten" der Arithmetik vorberechnet und in einer Tabelle gespeichert werden. Alles was man dann zur Rekonstruktion einer Zahl braucht sind* n *Multiplikationen und* $n - 1$ *Additionen. Das führt insbesondere zu einem relativ „billigen" Verfahren zur Berechnung des multiplikativen Inversen in* $\mathbb{Z}_m$.

4. *Das Verfahren funktioniert, wie man sich leicht überlegt, nicht nur für ganze Zahlen, sondern für beliebige euklidische Ringe modulo Elementen, die in disjunkte irreduzible Faktoren zerlegt werden können.*

5. *Der Name „CRT–Algorithmus" ist aus (Gathen & Gerhard, 1999) geborgt und stammt*[54] *daher, daß er den* Chinesischen Restsatz *(„Chinese Remainder Theorem") realisiert.*

Satz 2.38 (Chinesischer Restsatz) *Seien* $p_1, \ldots, p_n$ *paarweise teilerfremde Elemente des euklidischen Rings* R *und sei* $m = p_1 \cdots p_n$. *Dann ist*

$$R/\langle m \rangle \simeq R/\langle p_1 \rangle \times \cdots \times R/\langle p_n \rangle \tag{2.30}$$

und

$$(R/\langle m \rangle)^* \simeq (R/\langle p_1 \rangle)^* \times \cdots \times (R/\langle p_n \rangle)^* . \tag{2.31}$$

[54] Algebraiker haben es bestimmt schon erkannt.

Beweis: Sei wieder $m_j = m/p_j = \prod_{k\neq j} p_k$ und

$$\ell_j := \left(\left(m_j^{-1} \right)_{p_j} m_j \right)_m , \qquad j = 1, \ldots, n.$$

Da für $q, q' \in R$

$$q \equiv_m q' \qquad \Leftrightarrow \qquad m \,|\, (q - q') \qquad \Leftrightarrow \qquad p_j \,|\, (q - q') , \quad j = 1, \ldots, n,$$

gilt für die lineare Abbildung

$$L : \left\{ \begin{array}{rcl} R & \to & R/\langle p_1 \rangle \times \cdots \times R/\langle p_n \rangle, \\ q & \mapsto & (r_1, \ldots, r_n) = ((q)_{p_1}, \ldots, (q)_{p_n}), \end{array} \right.$$

daß $\ker L = m\,R = \langle m \rangle$. Die Surjektivität von L liefert uns eine enstprechende Varainte von Algorithmus 2.36, die uns zu jedem Satz von Resten r_j, $j = 1, \ldots, n$, ein Element $q \in R$, nämlich $q = \sum_{j=1}^{n} r_j \, \ell_j$, liefert, so daß $q \equiv_{p_j} r_j$, $j = 1, \ldots, n$. Damit gilt (2.30) und L ist der gewünschte Homomorphismus. □

Übung 2.12 Formulieren Sie den CRT–Algorithmus für beliebige euklidische Ringe.

Bemerkung 2.39 *Der Bezug zwischen Chinesischem Restsatz und Polynominterpolation wird am deutlichsten, wenn wir* $R = \mathbb{R}[x]$ *und* $p_j = p_j(x) = (x - x_j)$, $j = 0, \ldots, n$, *für paarweise verschiedene* $x_j \in \mathbb{R}$ *wählen, denn dann ist offensichtlich*[55]

$$\mathrm{ggT}\,(p_j, p_k) = \delta_{jk}\,(x - x_j) + (1 - \delta_{jk}) , \qquad j, k = 0, \ldots, n,$$

also entweder 1 *oder der gesamte Linearfaktor. Da*

$$m(x) = \prod_{j=0}^{n} (x - x_j) ,$$

ist $\mathbb{R}[x]/\langle m \rangle \simeq \Pi_n$. *Auf der anderen Seite ist* $\mathbb{R}[x]/\langle p_j \rangle = \mathbb{R}$, $j = 0, \ldots, n$, *da wir jedes* $f \in \mathbb{R}[x]$ *als*

$$f(x) = f\,(x_j) + (x - x_j)\, g(x), \qquad g(x) = \frac{f(x) - f\,(x_j)}{x - x_j},$$

darstellen können ("synthetische Division"). Damit ist

$$L(f) = (f\,(x_0) , \ldots, f\,(x_n)) .$$

Außerdem ist

$$m_j(x) = \prod_{k\neq j} (x - x_k) \qquad \Rightarrow \qquad (m_j)_{p_j} = m_j\,(x_j) = \prod_{k\neq j} (x_j - x_k) \neq 0,$$

was in $\mathbb{R} = \mathbb{R}[x]/\langle x - x_j \rangle$ *invertierbar ist und somit sind die "Rekonstruktionselemente" im Ring* R *gerade*

$$\ell_j(x) = \prod_{k\neq j} \frac{x - x_k}{x_j - x_k} \quad \in \Pi_n = \mathbb{R}[x]/\langle m \rangle.$$

[55] Und nach Normalisierung!

Mit Hilfe des chinesischen Restsatzes können wir noch ein wenig elementare Zahlentheorie betreiben und uns zuerst einmal die **Euler–Funktion** aus Definition 2.30 genauer ansehen.

Satz 2.40 (Euler–Funktion) *Seien* $m, m' \in \mathbb{N}$.

1. *Ist* $\mathrm{ggT}(m, m') = 1$, *dann ist* $\varphi(m, m') = \varphi(m)\,\varphi(m')$.
2. *Hat* m *die Primfaktorlegung* $m = \prod_j p_j^{e_j}$, *dann ist*

$$\varphi(m) = \prod_{j=1}^{n} p_j^{e_j-1}(p_j - 1). \tag{2.32}$$

Beweis: 1) ist nur eine Umformulierung des chinesischen Restsatzes für einen Spezialfall: Ist $\mathrm{ggT}(m, m') = 1$, dann ist $\mathbb{Z}_{mm'} \simeq \mathbb{Z}_m \times \mathbb{Z}_{m'}$ und

$$\mathbb{Z}_m^\times \ni a \simeq ((a)_m, (a)_{m'}) \in \mathbb{Z}_m^\times \times \mathbb{Z}_{m'}^\times$$

sagt uns, daß a genau dann eine Einheit ist, wenn $(a)_m$ und $(a)_{m'}$ Einheiten sind. Davon gibt es aber gerade $\varphi(m)\varphi(m')$ Stück.

Da Primzahlen von Haus aus teilerfremd sind, können wir uns dank 1) beim Beweis von 2) auf den Fall $n = 1$ beschränken. Nun ist $a \in \mathbb{Z}_{p^e}$ aber genau dann eine Einheit, wenn[56]

$$\mathrm{ggT}(a, p^e) = 1 \qquad \Leftrightarrow \qquad \mathrm{ggT}(a, p) = 1$$

ist, bzw. eine Nicht–Einheit[57], wenn $a \in p\mathbb{Z}_{p^{e-1}}$ ist. Dieser Ring hat aber p^{e-1} Elemente, weswegen

$$\varphi(p^e) = \#\mathbb{Z}_{p^e}^\times = \#\mathbb{Z}_{p^e} - \#\mathbb{Z}_{p^{e-1}} = p^e - p^{e-1} = p^{e-1}(p-1)$$

sein muss. □

Aus diesen Beobachtungen ergibt sich eine sehr nützliche Formel.

Satz 2.41 (Satz von Gauß) *Für* $m \in \mathbb{N}$ *und* $a \in \mathbb{Z}_m^\times$ *gilt*

$$a^{\varphi(m)} \equiv_m 1. \tag{2.33}$$

Beweis: Für $m = 1$ ist $\mathbb{Z}_m = \{0\}$, also $\mathbb{Z}_m^\times = \emptyset$ und damit ist die Behauptung *trivialerweise*[58] erfüllt. Ansonsten ist $\mathbb{Z}_m^\times$ eine (endliche) **multiplikative Gruppe** mit $\varphi(m)$ Elementen. Dann muss es aber einen Index k geben, daß $a^k = 1$

[56] *Jeder* gemeinsame Teiler von a und p^e muss von der Form p^k, $k < e$, sein, damit ist aber immer auch p ein Teiler von a, wenn $\mathrm{ggT}(a, p^e) \neq 1$ ist.

[57] Achtung: $\mathbb{Z}_{p^e}$ ist *nicht* $\mathbb{F}_{p^e}$, denn der Körper hat nur eine Nicht–Einheit, nämlich das Nullelement.

[58] Für die Elemente der leeren Menge sind, als Grundsatz der formalen Logik, *alle* Aussagen korrekt, sie sind beispielsweise gleichzeitig gerade und ungerade.

ist, denn die Folge $(a^j \ : \ j \in \mathbb{N})$ kann nur endlich viele verschiedene Werte annehmen, so daß es $1 \le j < j' \le m$ gibt, so daß

$$a^j = a^{j'} \qquad \Leftrightarrow \qquad 1 = a^{j'-j} =: a^k, \qquad k \in \mathbb{Z}_m.$$

Wenn es solche Idizes gibt, dann gibt es aber auch einen *kleinsten* derartigen Index $k(a)$, den man **Ordnung** des Gruppenelements nennt. Mit der **Untergruppe**

$$U_a = \left\{ a^k \ : \ 0 \le k < k(a) \right\}$$

erhalten wir, daß für $b, b' \in \mathbb{Z}_m^\times$ entweder

$$b\,U_a = b'U_a \qquad \Leftrightarrow \qquad ba^k = b'a^{k'} \qquad \Leftrightarrow \qquad b = b'a^{k'-k} \in b'\,U_a$$

oder

$$b\,U_a \cap b'U_a = \emptyset \qquad \Leftrightarrow \qquad b \notin b'\,U_a$$

erfüllt sein muß, und es daher eine Teilmenge $B \subset \mathbb{Z}_m^\times$ geben muss, so daß $\mathbb{Z}_m^\times$ die *disjunkte* Zerlegung

$$\mathbb{Z}_m^\times = \bigcup_{b\in B} b\,U_a \qquad \Rightarrow \qquad \varphi(m) = \#\mathbb{Z}_m^\times = \#B\,\#U_a = \#B\,k(a)$$

liefert. Also gilt $k(a)|\varphi(m)$ für jedes $a \in \mathbb{Z}_m^\times$ und daher auch

$$a^{\varphi(m)} \equiv_m a^{\#B\,k(a)} \equiv_m (\underbrace{a^{k(a)}}_{=1})^{\#B} \equiv_m 1,$$

wie behauptet. □

Übung 2.13 Zeigen Sie: Mit den Bezeichnungen aus Satz 2.41 gilt

$$a^k \equiv_m 1 \qquad \Leftrightarrow \qquad k(a)|k.$$

♢

Korollar 2.42 (Kleiner Fermat) *Für eine Primzahl* p *und* $a \in \mathbb{Z} \setminus p\mathbb{Z}$ *gilt*

$$a^{p-1} \equiv_p 1. \tag{2.34}$$

Beweis: Man wendet Satz 2.41 auf $(a)_p \in \mathbb{Z}_p$ an und beachtet, daß $\varphi(p) = p - 1$ ist, siehe (2.32). □

And now I must stop saying what I am not writing about, because there's nothing so special about that; every story one chooses to tell is a kind of censorship, it prevents the telling of other tales ...

S. Rushdie, *Shame*

3 Grundlegende kryptographische Verfahren

Jetzt, wo es mit der eigentlichen Kryptographie losgeht, erinnern wir uns noch einmal an unsere Terminologie der *Klartextnachricht* M, des *Schlüssels* K und der *Chiffriertransformation* E sowie der *Dechiffriertransformation* D. Zusammen machen diese dann das **Kryptosystem** aus.

3.1 Perfekte Sicherheit und One–Time–Pads

Ein erster Ansatz zur Sicherheit von Verschlüsselungsalgorithmen ist das wahrscheinlichkeitstheoretisch fundierte Konzept der perfekten Sicherheit, das auf Shannon[59] zurückgeht.

Dazu betrachtet man die Wahrscheinlichkeitsverteilung der Klartexte und der verschlüsselten Texte. Dann ist die Wahrscheinlichkeit[60], daß ein bestimmter Klartext M auftritt $p(M)$ und die *bedingte* Wahrscheinlichkeit, daß ein bestimmter Klartext M aus einem Chiffretext C gewonnen wurde, $p(M|C)$.

Definition 3.1 (Shannon) *Ein Kryptosystem heißt **perfekt sicher**, wenn*

$$p(M|C) = p(M), \qquad C \in \mathscr{C}, \tag{3.1}$$

gilt.

Perfekte Sicherheit bedeutet also, daß sich selbst unter Kenntnis des Chiffretexts die Klartexte so verteilen wie ohne diese, man kann also vermittels des Chiffretexts den Suchraum in keinster Weise einschränken.

[59] Claude Shannon, Vater der Informations- und Codierungstheorie und eine der prägenden Figuren der elektronischen, insbesondere digitalen, Signalverarbeitung.

[60] Ein völlig unrealistisches Maß wäre Gleichverteilung, nicht viel besser ist es, Buchstabenhäufigkeiten zu verwenden und die Buchstaben als unabhängig anszusehen, so daß „*yzyz*“ zumindet unwahrscheinlicher ist als „*nene*“. Obwohl: Letzteres ist der häufigkeitsbasierte Ansatz von Poe, der dann allerdings versucht, weitere Buchstaben durch das Vervollständigen von Wörtern zu bilden.

Bemerkung 3.2 *Dieses Sicherheitskonzept basiert auf Ideen der* ***Informationstheorie****, die ihrerseits wieder sehr wahrscheinlichkeitstheoretisch geprägt ist. Dort gibt es das Konzept der* ***punktweisen gegenseitigen Information***[61]*, zweier Ereignisse, die als*

$$\phi(x,y) := \log \frac{p(x,y)}{p(x)\,p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)}$$

definiert. Die ***gegenseitige Information*** *ist dann der Erwartungswert*

$$I(X,Y) = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\,\varphi(x,y) = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\,\log\frac{p(x|y)}{p(x)}.$$

Daß hier Logarithmen auftreten, ist in sehr sinnvoll und in keinster Weise willkürlich[62] *und basiert im wesentlichen auf dem Konzept der* ***Entropie****, siehe Für ein perfekt sicheres Kryptosystem bedeutet das dann, daß*

$$I(\mathscr{M},\mathscr{C}) = -\sum_{M\in\mathscr{M}}\sum_{C\in\mathscr{C}} p(M,C)\,\overbrace{\log\underbrace{\frac{p(M|C)}{p(M)}}_{=1}}^{=0} = 0$$

sein muss: Es gibt keine gemeinsame Information zwischen Chiffre und Klartext. *Und damit kann man auch keinen Zusammenhang finden . . .*

Perfekte Sicherheit hat ihre Konsequenzen. Dazu nehmen wir an, daß[63]

$$p(M) > 0, \quad M\in\mathscr{M}, \qquad \sum_{M\in\mathscr{M}} p(M) = 1. \tag{3.2}$$

Da DE die Identität sein soll, muss $E : \mathscr{M} \to \mathscr{C}$ **injektiv** sein, und damit ist auch $\#\mathscr{C} \geq \#\mathscr{M}$. Ist nun das System perfekt sicher und $C \in \mathscr{C}$, dann ist

$$0 < p(M) = p(M|C), \qquad M\in\mathscr{M},$$

so daß es zu jedem Chiffretext C einen Schlüssel K geben muss, so daß $M = D(K,C)$ ist, denn andernfalls wäre ja $p(M|C) = 0$. Für festes $M \in \mathscr{M}$ ist daher die Abbildung $K \mapsto E(K,M)$ **surjektiv** auf $\mathscr{C}$, also

$$\#\mathscr{K} \geq \#\mathscr{C} \geq \#\mathscr{M}. \tag{3.3}$$

Mit anderen Worten: *Für perfekte Sicherheit brauchen wir mindestens so viele verschiedene Schlüssel wie Klartextbotschaften.*

Beispiel 3.3 (Vernam One Time Pad) *Das einfachste*[64] *Beispiel eines sicheren Kryptosystems ist das (****Vernam) One Time Pad****, das von G. Vernam 1918 zum Patent*[65]

[61] Auf Englisch **pointwise mutual information**.
[62] Im Gegensatz zu dB–Skala.
[63] Wer interessiert sich auch für Klartexte, die es nicht gibt?
[64] Und anscheinend auch einzige.
[65] US Patent Nr. US1310719A

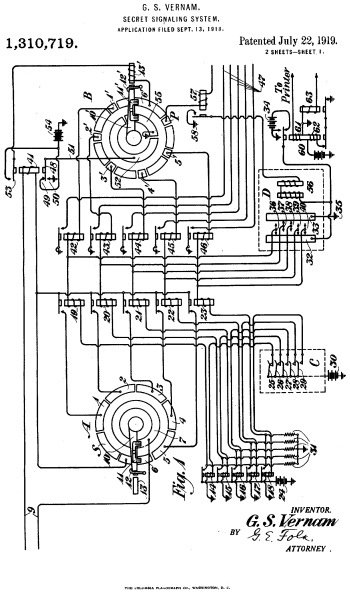

G. S. VERNAM.
SECRET SIGNALING SYSTEM.
APPLICATION FILED SEPT. 13, 1919.
1,310,719. Patented July 22, 1919.
2 SHEETS—SHEET 1.
Fig. 1
INVENTOR.
G. S. Vernam
BY
ATTORNEY.

Abbildung 3.1: Auszug aus Vernams Patentschrift mit der Verdrahtung des One Time Pad. *Quelle: Google*

angemeldet und 1919 erteilt wurde, siehe Abb. 3.1. Mathematisch verwendet man ein Alphabet $A \simeq \mathbb{Z}_m$, $\mathscr{M} = \mathscr{C} = \mathscr{K} = \mathbb{Z}_m^n$ *und die Abbildungen*

$$E(K, M) = M \oplus K := ((m_j + k_j)_m \,:\, j = 1, \ldots, n)\,, \tag{3.4}$$
$$D(K, C) = C \ominus K := ((c_j - k_j)_m \,:\, j = 1, \ldots, n)\,. \tag{3.5}$$

So einfach das One Time Pad auch ist, es ist perfekt sicher.

Proposition 3.4 *Sind alle Schlüssel gleichwahrscheinlich, also* $p(K) = m^{-n}$, $K \in \mathscr{K}$, *dann ist das One Time Pad perfekt sicher.*

Beweis: Zu jedem $M \in \mathscr{M}$ und $C \in \mathscr{C}$ gibt es genau ein K mit

$$C = E(K, M) = K \oplus M \qquad \text{nämlich} \qquad K = C \ominus M,$$

und damit ist[66] $p(C, M) = p(M)\,p(K) = m^{-n}\,p(M)$ und somit

$$p(C) = \sum_{M \in \mathscr{M}} p(C, M) = \sum_{M \in \mathscr{M}} m^{-n}\,p(M) = m^{-n} \underbrace{\sum_{M \in \mathscr{M}} p(M)}_{=1} = m^{-n}.$$

[66] Der Schlüssel ist ja **unabhängig** vom Klartext oder sollte es zumindest besser sein.

Ds liefert aber die bedingte Wahrscheinlichkeit

$$p(M|C) = \frac{p(M,C)}{P(C)} = \frac{m^{-n}\, p(M)}{m^{-n}} = p(M), \qquad M \in \mathscr{M},\ C \in \mathscr{C},$$

und damit ist das One Time Pad perfekt sicher. □

Bemerkung 3.5 *Bei der praktischen Realisierung eines One Time Pad*[67] *sind zwei wichtige Probleme zu lösen:*

1. *Perfekte Zufälligkeit der Schlüssel ohne irgendwelche Präferenzen des Pseudozufallszahlengenerators. Die (computergestützte) Bestimmung wirklicher Zufallsfolgen ist nämlich ein hochgradig nichttrivialer Prozess.*

2. *Sichere Übermittlung und Aufbewahrung des Schlüssels.*

Beides erfordert in der Praxis einen gewissen Aufwand, sehr unterhaltsam beschrieben in (Stephenson, 1999).

3.2 Ein symmetrisches Verfahren: AES

Als nächstes sehen wir uns ein **symmetrisches Verschlüsselungsverfahren** an, und zwar das **AES**–Verfahren, den Nachfolger des **DES**–Standards. Dabei steht „DES" für *„**D**ata **E**ncryption **S**tandard"*, ein Verfahren aus den siebziger Jahren[68], das wegen seiner geringen Schlüssellänge als nicht mehr sicher galt und 2001 durch den **AES** (*„**A**dvanced **E**ncryption **S**tandard"*) ersetzt wurde, siehe (Willems, 2008).

Bei diesem Verschlüsselungsverfahren arbeitet man komplett **binär**, das heißt, in $\mathbb{F}_2$ und geht davon aus, daß ein Schlüssel $K \in \mathbb{F}_2^n$ bereits ausgetauscht ist, wobei für die **Blocklänge** n die Möglichkeiten $n \in \{128, 192, 256\}$ zulässig sind. Die Nachricht $M \in \mathbb{F}_2^N$ wird dann in Blöcke der Länge n zerlegt[69], die separat behandelt werden, so daß wir auch gleich davon ausgehen können, daß $M \in \mathbb{F}_2^n$ ist.

Mit dem irreduziblen Polynom

$$f(x) = x^8 + x^4 + x^3 + x + 1 \in \mathbb{F}_2[x] \tag{3.6}$$

kann man nach Proposition 2.34 den endlichen Körper

$$\mathbb{F}_{2^8} = \mathbb{F}_2[x]/\langle f\rangle \simeq \Pi_7 \simeq \mathbb{F}_2^8$$

bilden und damit jedes Körperelement wahlweise mit dem Polynom $g(x) = a_7x^7 + \cdots + a_0 \in \Pi_7(\mathbb{F}_2)$ oder mit dem **Byte** $a = a_7 \ldots a_0 \in \mathbb{Z}_{256} \simeq \mathbb{F}_{2^8}$ identifizieren. Der Block der Länge n zerfällt dann in $n/8 = \{16, 24, 32\}$ Bytes;

[67] Angeblich ist die Kommunikation zwischen dem Weißen Haus und dem Kreml mit einem One Time Pad verschlüsselt.

[68] ... des letzten Jahrtausends.

[69] Ensteht dabei ein Block der Länge $< n$, so wird dieser einfach mit 0 aufgefüllt

beschränken wir uns auf den Fall $n = 128$, dann sind das also 16 Bytes der Nachricht $M = m_{00}m_{10}\dots m_{33} \in \mathbb{F}_2^{128}$, die wir *spaltenweise* in einer Matrix

$$M = \begin{bmatrix} m_{00} & \dots & m_{03} \\ \vdots & \ddots & \vdots \\ m_{30} & \dots & m_{33} \end{bmatrix} =: [M_0 \dots M_3] \in \mathbb{F}_{2^8}^{4\times 4}$$

anordnen können, die wir genauso nennen wie die Nachricht M selbst[70]. Ausserdem setzen wir $K^{(0)} = K$, wobei K der Schlüssel ist, den wir ebenfalls wahlweise als Bitfolge in $\mathbb{F}_2^{128}$ oder als Matrix in $\mathbb{F}_{2^8}^{4\times 4}$ betrachten.

Jetzt verschlüsselt man durch einfache Addition mit dem Schlüssel und stiftet dann in der Matrix M Verwirrung, indem man für $j = 0, \dots, 9$, also *zehnmal*, die folgenden Operationen durchführt:

□ `Addition des Rundenschlüssels`: Zur Matrix M wird der **Rundenschlüssel** $K^{(j)}$ addiert:

$$M \leftarrow M + K^{(j)} \qquad \text{in } \mathbb{F}_{2^8}^{4\times 4}, \tag{3.7}$$

wobei jede Addition in $\mathbb{F}_{2^8}$ durchzuführen ist[71]. Das kann man natürlich durch Subtraktion wieder rückgängig machen

□ `SubBytes`: Für jedes der Bytes $a = a_7 \dots a_0$ aus M berechnet man $a' = g(f(a))$ mit der *nichtlinearen* Abbildung

$$f(x) = \begin{cases} x^{-1}, & x \neq 0, \\ 0, & x = 0, \end{cases} \qquad x \in \mathbb{F}_{2^8} \tag{3.8}$$

und der linearen Transformation

$$g(a) = Ga + b := \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_7 \\ \vdots \\ a_0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}. \tag{3.9}$$

Die **zirkuläre Matrix** G ist von der Block–Form

$$G = \begin{bmatrix} R & R^T \\ R^T & R \end{bmatrix}, \qquad R = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

und erfüllt[72] $\det G = 1$, so daß G eine invertierbare Matrix ist. Das heißt,

$$M \leftarrow g\left(f(M)\right), \tag{3.10}$$

[70] Ist ja auch nichts anderes, nur eine Umordnung.

[71] Also als 8–Tupel in $\mathbb{F}_2$.

[72] Nachrechnen! Über $\mathbb{R}$ gilt $\det G = 5$.

wobei die Operation von f und g **komponentenweise** zu verstehen ist.
□ `ShiftRows`: In der Matrix werden die *Zeilen* zyklisch geschoben, was

$$M \leftarrow \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{11} & m_{12} & m_{13} & m_{10} \\ m_{22} & m_{23} & m_{20} & m_{21} \\ m_{33} & m_{30} & m_{31} & m_{32} \end{bmatrix} \tag{3.11}$$

liefert und natürlich ebenfalls eine invertierbare Operation ist.
□ `MixColumns`: Jetzt werden auch noch die Spalten durcheinandergemischt:

$$M \leftarrow TM, \qquad T := \begin{bmatrix} a & b & 1 & 1 \\ 1 & a & b & 1 \\ 1 & 1 & a & 1 \\ b & 1 & 1 & a \end{bmatrix}, \qquad a = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \; b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \in \mathbb{F}_{2^8}. \tag{3.12}$$

Auch die Matrix T ist invertierbar, es empfiehlt sich aber, ein Computeralgebrasystem wie beispielsweise `Maple` oder `Maxima` zu verwenden, um das zu verifizieren.
□ `Update des Rundenschlüssels und Inkrement von` j: Der nächste Rundenschlüssel $K^{(j+1)} = \left[K_0^{(j+1)} \dots K_3^{(j+1)}\right]$ ergibt sich als

$$K_0^{(j+1)} = K_0^{(j)} + H\left(K_3^{(j)}\right) \tag{3.13}$$

$$K_k^{(j+1)} = K_k^{(j)} + K_{k-1}^{(j+1)}, \qquad k = 1, 2, 3, \tag{3.14}$$

mit

$$H(x) := g(f(x)) + \begin{bmatrix} a^{j-1} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

und den Funktionen f, g aus (3.8) und (3.9) und dem Wert a aus (3.12). Da die Rundenschlüssel von der Nachricht M unabhängig sind, kann man mit den Regeln (3.13) und (3.14) auch die Matrix

$$K' = \left[K_0^{(0)} \dots K_3^{(0)} K_0^{(1)} \dots K_3^{(1)} \dots K_0^{(9)} \dots K_3^{(9)}\right]$$

als „erweiterten Schlüssel“ vorberechnen.

Ein kleines Detail ist allerdings hier unterschlagen worden: Im *letzten* Durchgang ($j = 9$) lässt man die Prozedur `MixColumns` weg, da sie da keinen wirklichen Sicherheitsgewinn mehr bringt.

Ein wesentlicher Punkt in der Prozedur `SubBytes` ist die Wahl der *nichtlinearen* Funktion f aus (3.8), die dazu dient den Schlüssel zu verschleiern, der ja *für jeden* Verschlüsselungsvorgang verwendet wird. Würde man nämlich lediglich

den Schlüssel addieren, aber *kein* One Time Pad verwenden, dann könnte man aus der Kenntnis eines „Klartextschnipsels“ $m \in \mathbb{F}_{2^n}$ durch einfache Subtraktion vom Chiffretext $c \in \mathbb{F}_{2^n}$ den Schlüssel $k \in \mathbb{F}_{2^n}$ ermitteln:

$$c = m + k \qquad \Leftrightarrow \qquad k = c - m.$$

Dies kann auch eine lineare Funktion ℓ nicht verschleiern:

$$\ell(c) = \ell(m + k) = \ell(m) + \ell(k) \qquad \Rightarrow \qquad \ell(k) = \ell(c) - \ell(m)$$

und $\ell(k)$ ist alles, was wir zum Verschlüsseln brauchen. Man kann das auch anders sehen, denn im Fall einer linearen Funktion ist

$$\#\{\ell(x + k) - \ell(x) \ : \ x \in \mathbb{F}_{2^n}\} = \#\{\ell(k)\} = 1$$

minimal. Besser sind Funktionen $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$, bei denen

$$N_f := \{f(x + k) - f(x) \ : \ x \in \mathbb{F}_{2^n}\}$$

für $k \neq 0$ möglichst groß wird und man kann zeigen, daß für ungerades n die Funktion aus (3.8) hier wirklich optimal ist, siehe (Willems, 2008).

3.3 Exkurs: Wann ist ein Polynom irreduizbel?

Einen klitzekleinen Punkt haben wir im letzten Abschnitt unterschlagen: Wir kommt man eigentlich darauf, daß das Polynom f aus (3.6) wirklich irreduzibel ist? Oder, allgemeiner, wie kann man das für ein beliebiges Polynom überprüfen und gibt es Möglichkeiten, irreduzible Polynome *beliebiger Ordnung* beispielsweise über $\mathbb{F}_2$ zu konstruieren. Man könnte hier beispielsweise an ein „AES für Fortgeschrittene“ über dem größeren Körper $\mathbb{F}_{2^{16}}$ denken ...

Wir werden ein Irreduzibilitätskriterium[73] kennenlernen, das auf dem kleinen Fermat, Korollar 2.42, basiert. Dazu sei $q = p^n$ eine **Primzahlpotenz** und $\mathbb{F}_q$ der zugehörige endliche Körper gemäß Proposition 2.34. Auch wenn man die irreduziblen Polynome nur schwer auflisten kann, geht das sehr gut mit ihrem Produkt.

Satz 3.6 *Für $d \geq 1$ ist das Polynom $f^*(x) = x^{q^d} - x \in \mathbb{F}_q[x]$ das Produkt aller irreduziblen Polynome in $\mathbb{F}_q[x]$, deren Grad d teilt.*

Zuerst benötigen wir eine Erweiterung des kleinen Fermat.

Lemma 3.7 *Ist q eine Primzahlpotenz, dann gilt in $\mathbb{F}_q[x]$ die Identität*

$$x^q - x = \prod_{a \in \mathbb{F}_q} (x - a). \tag{3.15}$$

[73] Aus (Gathen & Gerhard, 1999).

Beweis: Nach (2.34) ist $a^q \equiv_q a$ für alle $a \in \mathbb{F}_q$ und damit ist *jedes* $a \in \mathbb{F}_q$ eine Nullstelle von $f^*(x) := x^q - x$. Nach Division mit Rest ist dann aber

$$f^*(x) = (x-a)g_a(x) + \underbrace{f^*(a)}_{=0} = (x-a)g_a(x) \qquad \Rightarrow \qquad x-a | f^*(x), \quad a \in \mathbb{F}_q,$$

und da für $a \neq b$ ja $\mathrm{ggT}\,(x-a, x-b) = 1$ gilt, muß (3.15) erfüllt sein. □

Beweis von Satz 3.6: Da $q^d = p^{nd}$ ebenfalls eine Primzahlpotenz ist, gilt (3.15) in $\mathbb{F}_{q^d}$, also

$$x^{q^d} - x = \prod_{a \in \mathbb{F}_{q^d}} (x-a) \qquad \text{in } \mathbb{F}_{q^d}. \tag{3.16}$$

Gäbe es ein[74] $g \in \mathbb{F}_q[x] \setminus \mathbb{F}_q$, mit $g^2 | f^*$, dann muss es wegen der Faktorisierung (3.16) mindestens ein $a \in \mathbb{F}_{q^d}$ geben mit $(x-a)|g$ und damit den Widerspruch $(x-a)^2 | f^*$. Also ist f^* **quadratfrei**.

Nehmen wir nun an, $f \in \mathbb{F}_q[x]$ sei ein irreduzibles **monisches Polynom**[75] und zeigen wir daß dann

$$f \mid f^* \quad \Leftrightarrow \quad \deg f \mid \deg f^* \tag{3.17}$$

gelten muss.

Im Falle $f | f^*$ betrachten wir die **Körpererweiterung**[76] $\mathbb{F}_q \subseteq \mathbb{F}_{q^d}$. Nun nutzen wir wieder (3.16) und erhalten, daß es $A \subset \mathbb{F}_{q^d}$ geben muss, so daß

$$f(x) = \prod_{a \in A} (x-a) \qquad \text{in } \mathbb{F}_{q^d}[x]$$

gilt. Weil f irreduzibel ist, ist $\mathbb{F}_f := \mathbb{F}_q[x]/\langle f \rangle \subseteq \mathbb{F}_{q^d}$ ein **Teilkörper**[77] von $\mathbb{F}_{q^d}$ mit $q^{\deg f}$ Elementen. Damit muss aber $q^d = (\#\mathbb{F}_f)^k = q^{k \deg f}$ für passendes k sein, also $\deg f | d = \deg f^*$.

Nehmen wir umgekehrt $m := \deg f | \deg f^* =: d$ an und betrachten wieder den Körper $\mathbb{F}_{q^m} = \mathbb{F}_q[x]/\langle f \rangle$ und sei $a := (x)_f \in \mathbb{F}_{q^m}$ eine Wurzel von f. Der kleine Fermat impliziert dann $a^{q^m} = a$. Da[78]

$$(q^m - 1) \sum_{j=1}^{d/m} q^{d-jm} = \sum_{j=1}^{d/m} q^{d-(j-1)} - \sum_{j=1}^{d/m} q^{d-jm} = \sum_{j=0}^{d/m-1} q^{d-jm} - \sum_{j=1}^{d/m} q^{d-jm} = q^d - 1,$$

also $q^d - 1 = (q^m - 1)\, r(q)$, ist auch

$$x^{q^d-1} - 1 = \left(x^{q^m-1} - 1\right) \sum_{j=0}^{r(q)-1} x^{j(q^m-1)}. \tag{3.18}$$

[74] Also ein *nichtkonstantes* Polynom.

[75] Also ein Polynom der Form $f(x) = x^k + \cdots$.

[76] Wenn wir uns an Proposition 2.34 erinnern, so sagt uns die Konstruktion mit dem irrduziblen Polynom ja, daß $\mathbb{F}_q$ gerade mit den konstanten Polynomen in $\mathbb{F}_{q^d} = \mathbb{F}_q[x]/\langle g \rangle$ identifiziert werden kann, was einer **Einbettung** von $\mathbb{F}_q$ in $\mathbb{F}_{q^d}$ entspricht.

[77] Eine Teilmenge, die selbst wieder ein Körper, also insbesondere abgeschlossen unter den arithmetischen Operationen ist.

[78] Der Standardtrick der geometrischen Reihe ...

Multiplizieren wir beide Seiten von (3.18) mit x, so erhalten wir

$$\left(x^{q^m} - x\right) \mid \left(x^{q^d} - x\right)$$

und da a eine Nullstelle von $x^{q^m} - x$ ist, gilt auch

$$(x - a) \mid \left(x^{q^m} - x\right) \mid \left(x^{q^d} - x\right) \qquad \Rightarrow \qquad (x - a) \,|\, ggT\,(f, x^{q^d} - x),$$

all das in $\mathbb{F}_{q^m}[x]$. Damit ist $ggT\,(f, x^{q^d} - x) \neq 1$ in $\mathbb{F}_q[x]$ und da f irreduzibel ist, kann nur $ggT\,(f, x^{q^d} - x) = f$ sein, also $f \,|\, x^{q^d} - x$. □

Übung 3.1 Beweisen Sie (3.18). ◇

Was nützt uns nun dieses Resultat? In der Tat liefert es nach einer einfachen Umformulierung einen Test auf Irreduzibilität.

Korollar 3.8 (Irreduzibilitätstest für Polynome) *Ein Polynom* $f \in \mathbb{F}_q[x]$ *mit* $\deg f \geq 1$ *ist genau dann irreduzibel, wenn*

1. $f \,|\, x^{q^{\deg f}} - x$,
2. $ggT\left(f, x^{q^{\deg f/k}} - x\right) = 1$ *für jeden* ***Primfaktor*** k *von* $\deg f$.

Beweis: Ist f irreduzibel, dann teilt es natürlich das f^* aus Satz 3.6 für alle Vielfachen d von $\deg f$, insbesondere für $\deg f$ selbst, was 1) liefert. Und natürlich ist $ggT\,(f, g) = 1$ für jedes g mit $\deg g < \deg f$, weswegen auch 2) erfüllt sein muss.

Interessant ist also nur die Umkehrung. Seien 1) und 2) erfüllt und sei g ein echter irreduzibler Faktor von f, dann folgt, mit 1), daß $g \,|\, f \,|\, x^{q^{\deg f}} - x$, dann gilt nach Satz 3.6 auch $\deg g \,|\, \deg f$. Damit ist $\deg f = kk' \deg g$, wobei k eine Primzahl ist und $g \,|\, x^{k' \deg g} - x = x^{\deg f/k} - x$, wieder nach Satz 3.6, denn schließlich ist g ja irreduzibel. Das ist aber ein Widerspruch zu 2), weswegen f irreduzbibel sein muss. □

Damit können wir den Irreduzibilitätstest auf ggT –Berechnungen und die Überprüfung

$$ggT\left(f, x^{q^{\deg f}} - x\right) = f, \qquad ggT\left(f, x^{q^{\deg f/p}} - x\right) = 1, \quad p \,|\, \deg f, \qquad \text{in } \mathbb{F}_q[x] \tag{3.19}$$

zurückführen, die man mit einem geeigneten **Computeralgebrasystem**[79] durchführen kann.

Beispiel 3.9 (Irreduzibilitätstest) *Wir testen die Irreduzibilität unseres Polynoms aus* (3.6) *mit Hilfe des OpenSource–CAS* **`Maxima`**, *siehe (Schelter, 2001). Für* `Maxima` *gibt es ein Paket „*`gf`*" (für* **G***alois* **F***ield), das spätestens ab Version 5.30 auch in den Sourcecode integriert ist. Das Ganze läuft recht einfach und entspannt ab. Zuerst setzt man seinen Grundköper auf* $\mathbb{F}_2 = \mathbb{F}_2/\langle x \rangle$*:*

[79] Abegkürzt **CAS**.

```
(%i1) gf_set_data( 2,1 );
(%o1) gf_data(characteristic = 2, exponent = 1, reduction = x,
              primitive = 1, cardinality = 2, order = 1,
              factors_of_order = [])
```

Der Funktion `gf_set_data` *wird dabei immer die Primzahl des Grundkörpers, in unserem Fall also* 2 *sowie entweder ein irreduzibles Polynom vom Grad* ≥ 1 *oder dessen Grad, hier die* 1*, übergeben. Im letzteren Fall sucht sich* `Maxima` *das irreduzible Polynom selbst. Wenn das einmal geklärt ist, brauchen wir nur noch die beiden* ggT*–Berechnungen*

```
(%i2) gf_gcd( x^8 + x^4+x^3+x+1,x^256 - x );
                                8    4    3
(%o2)                          x  + x  + x  + x + 1
```

und

```
(%i3) gf_gcd( x^8 + x^4+x^3+x+1,x^16 - x );
(%o3)                                  1
```

durchführen zu lassen und schon wissen wir, daß unser Polynom irreduzibel ist.

Bemerkung 3.10 *Die Irreduzibilität eines Polynoms hängt ganz massiv vom Grundkörper ab! So ist beispielsweise* $f(x) = x^2 + x + 1$ *über* $\mathbb{F}_2$ *ein irreduzibles Polynom, über* $\mathbb{F}_3$ *aber nicht.*

Übung 3.2 Verfizieren Sie via `Maxima`, daß die Funktion $f(x) = x^2+x+1$ über $\mathbb{F}_2$ irreduzibel ist, über $\mathbb{F}_3$ aber nicht und beweisen Sie dies, indem Sie zeigen, daß ein quadratisches Polynom f genau dann irreduzibel über $\mathbb{F}_q$ ist, wenn $0 \notin f(\mathbb{F}_q)$ ist. ◇

Bleibt also nur noch die Frage, wie man irreduzible Polynome von einem vorgegebenen Grad findet. Man könnte natürlich ausprobieren, denn schließlich ist $\#\Pi_n(\mathbb{F}_q) = q^n$, man muss also „nur" q^n verschiedene Polynome mit (3.19) testen. Etwas vornehmer bezeichnet man dieses Vorgehen als **Exhaustive Search**, was aber eben auch nichts anderes bedeutet als „Ausprobieren".

Etwas cleverere Verfahren verwenden hier den Zufall und wahrscheinlichkeitstheoretische Aussagen über den Anteil an irreduziblen Polynomen, um Vorhersagen zu machen, mit welcher Wahrscheinlichkeit man nach wie vielen Schritten ein geeignetes Polynom gefunden hat, siehe (Gathen & Gerhard, 1999). Und die ist relativ hoch, wie das folgenden Resultat, (Gathen & Gerhard, 1999, Lemma 14.38, S. 409) zeigt.

Lemma 3.11 *Die Anzahl* $I_{n,q}$ *der monischen irreduziblen Polynome vom Grad* n *in* $\mathbb{F}_q[x]$ *erfüllt*[80]

$$\frac{q^n - 2q^{n/2}}{n} \leq I_{n,q} \leq \frac{q^n}{n}.$$

Ist $q^n > 16$*, dann ist*

$$\frac{1}{2n} \leq \frac{1}{n}\left(1 - \frac{2}{q^{n/2}}\right) \leq p(I_{n,q}) \leq \frac{1}{n}.$$

[80] Nicht vergessen: Insgesamt gibt es q^n solche monischen Polynome.

3.4 Public–Key–Kryptographie und Einwegfunktionen

Ein wesentlicher Nachteil des AES–Verfahrens und seiner Derivate ist, daß zuerst einmal ein Schlüssel getauscht werden muss. Auch wenn man die im Rahmen von Schlüsseltauschparties sehr angenehm durchführen kann, setzt es eben einen sicheren Kanal voraus, über den die Schlüssel getauscht werden können, was in Zeiten des Internet nicht mehr sonderlich realistisch ist. Die Idee der Trennung von privaten und öffentlichen Schlüsseln stammt aus einer Arbeit von Diffie und Hellman(Diffie & Hellman, 1976) und setzt auf das Konzept der **Einwegfunktion**.

Definition 3.12 (Einwegfunktion) *Eine Funktion $f : X \to Y$ heißt **Einwegfunktion**, falls man $f(x)$ für jedes $x \in X$ einfach berechnen kann, $f^{-1}(y)$ für $y \in Y$ aber praktisch nicht.*

Bemerkung 3.13 *Der Begriff der Einwegfunktion ist immer mit einem gewissen Risiko verbunden! Normalerweise kann man nur zeigen, daß gewisse* bekannte *Algorithmen zur Besrechnung von $f^{-1}(y)$ eine gewisse Mindestkomplexität haben*[81]*. Zum anderen stellt sich die Frage, wie man Komplexität misst: Als **Worst–Case–Komplexität**, die sich nur dafür interessiert, wie unmöglich die Invertierung im schlimmsten Fall ist, oder als **Average–Case–Komplexität**, die sich nur für den Erwartungswert der Berechnungzeit interessiert und unter Umständen um Größenordnungen besser sein kann.*

Beispiel 3.14 (Einwegfunktionen) *Die bekanntesten und am häufigsten verwendeten Beispiele von Einwegfunktionen sind:*

1. ***Diskreter Logarithmus**: Ist $g \in G$ ein Gruppenelement der Ordnung n, d.h. $g^n = g$ und $g^k \neq g$, $k \in \mathbb{Z}_n$, dann ist die Abbildung*

$$\mathbb{Z}_n \ni x \mapsto g^x \in G_g = \left\{ g^k \,:\, k \in \mathbb{Z}_n \right\}$$

   eine Bijektion, deren Inverse man als diskreten Logarithmus bezeichnet und $x = \log_g y$, $y \in G_g$ schreibt. Für geeignete Gruppen und hinreichend große Werte von n gilt das als sichere Einwegfunktion, aber das werden wir uns noch genauer ansehen.

2. ***Primfaktorisierung**: Für große Zahlen ist die Berechnung einer Primfaktorzerlegung normalerweise praktisch nicht durchführbar. Kennen wir zwei große Primzahlen p, p', dann ist das Produkt $x = pp'$ relativ billig zu berechnen, die Rückgewinnung von p und p' aus x aber nur mit sehr großem Aufwand. Allerdings muss man die Primfaktoren jeweils als **sichere Primzahl** wählen, die mit gängigen Faktorisierungsverfahren nicht angreifbar sind. Um das zu verstehen, müssen wir uns aber auch ein wenig mit Faktorisierungsmethoden auseinandersetzen*[82]

Eine schöne Schilderung der Geschichte hinter den Kryptosystemen mit öffentlichen Schlüsseln findet sich wieder einmal in (Singh, 2000b).

[81] Man denke ans Multiplizieren von ganzen Zahlen: Das naive Verfahren hatte $O(N^2)$, das beste *bekannte* Verfahren aber $O(N \log N)$; dieses Verfahren ist aber erst seit (Schönhage & Strassen, 1971), also seit 1971, bekannt.

[82] Ja, das wird wieder mal Mathematik ...

3.5 Das RSA–Verfahren

Der Klassiker unter den Public–Key–Verfahren ist das **RSA–Verfahren**, das Rivest, Shamir und Adleman[83] 1978 vorgestellt und 1983 als Patent erteilt bekommen haben, (Rivest *et al.*, 1983), und das auf die Schwierigkeiten bei der Faktorisierung von Primzahlen vertraut.

Baustein dieses Verfahrens sind zwei **große Primzahlen** $p \neq p'$ und deren (noch größeres) Produkt $n = pp'$. Der Wert der **Euler–Funktion** φ aus (2.26) an der Stelle n ist dann $\varphi(n) = \varphi(p)\varphi(p') = (p-1)(p'-1)$. Schliesslich brauchen wir noch ein $e \in \mathbb{Z}_{\varphi(n)}$ mit der Eigenschaft[84] $\mathrm{ggT}\,(e, \varphi(n)) = 1$. Das fällt relativ leicht und lässt viele Möglichkeiten von e zu, wenn $p-1$ und $p'-1$ *nicht nur kleine* Primteiler haben. So etwas bezeichnet man als **sichere Primzahl**. Ideal sind hierbei (besonders gesuchte) Primzahlen der Form $p = 2q+1$, wobei q ebenfalls eine Primzahl ist. Nun bestimmt man noch

$$d \equiv_{\varphi(n)} e^{-1} \tag{3.20}$$

mit Hilfe des erweiterten euklidischen Algorithmus und hat alle Bausteine für das Verschlüsselungsverfahren beisammen.

Definition 3.15 (RSA–Schlüssel) ***Öffentlicher Schlüssel*** *des RSA–Verfahrens ist das Paar* $K_e := (n, e)$*, der* ***Privater Schlüssel*** *die Zahl* $K_d := d$*. Ein öffentlicher Schlüssel wird zu Codierungszwecken weitergegeben, der private Schlüssel dient der Entschlüsselung und muss sicher verwahrt werden.*

Man verwendet jetzt den Klartext- und Chiffreraum $\mathscr{M} = \mathscr{C} = \mathbb{Z}_n^N$ und da jede Komponente dieser langen Nachricht separat in $\mathbb{Z}_n$ ver- und entschlüsselt wird, können wir für die Beschreibung des Verfahrens auch gleich $N = 1$ annehmen. Für $m \in \mathbb{Z}_n$ ist nun die Verschlüsselung als

$$E(K, m) = E(K_e, m) := (m^e)_n, \qquad m \in \mathbb{Z}_n, \tag{3.21}$$

und die Entschlüsselung als

$$D(K, c) = D(K_d, c) := (c^d)_n, \qquad c \in \mathbb{Z}_n, \tag{3.22}$$

Die gute Nachricht zuerst: Das Verfahren tut, was es soll.

Satz 3.16

$$D(K, E(K, m)) = m, \qquad m \in \mathscr{M}. \tag{3.23}$$

Beweis: Wir müssen zeigen, daß für $x \in \mathbb{Z}_n$

$$x = \left((x^e)_n^d\right)_n = \left((x^e - nk)^d\right)_n = \left(\sum_{j=0}^{d} \binom{d}{j} x^{(d-j)e} (nk)^j\right)_n = \left(x^{ed}\right)_n,$$

[83] Wo da dann wohl der Name des Verfahrens herkommt?

[84] Das schliesst $e = 0$ automatisch aus.

also

$$x^{ed} \equiv_n x, \qquad x \in \mathbb{Z}_n, \tag{3.24}$$

erfüllt ist, was für $x = 0, 1$ ziemlich offensichtlich ist. Ist $x \in \mathbb{Z}_n^\times$, also $\mathrm{ggT}(x, p) = \mathrm{ggT}(x, p') = 1$, dann nutzen wir dank (3.20) daß $ed \equiv_{\varphi(n)} 1$, also $ed = k\,\varphi(n)+1$ für ein $k \in \mathbb{Z}$ ist, und Satz (2.33) ergibt

$$x^{ed} = x^{k\varphi(n)+1} = x\Big(\underbrace{x^{\varphi(n)}}_{\equiv_n 1}\Big)^k \equiv_n x,$$

also gerade (3.24) für $x \in \mathbb{Z}_n^\times$.

Bleiben also noch die Fälle $p|x$ oder $p'|x$, beides gleichzeitig ist aber nicht möglich, da dann $x \geq pp' = n$ wäre, also $x \notin \mathbb{Z}_n$. Aus Symmetriegründen können wir uns auf den ersten Fall, also $x = rp$, $r \in \mathbb{Z}_{p'}$, beschränken. Dann gilt

$$x^{ed} = x^{k\varphi(p)\varphi(p')+1} = x\left(x^{\varphi(p')}\right)^{k\varphi(p)} \equiv_{p'} x$$

und

$$p|x \qquad \Rightarrow \qquad p|x^{ed} - x,$$

also $n = pp'|x^{ed} - x$ und damit wieder $x^{ed} \equiv_n x$, also (3.24) für $x \in \mathbb{Z}_n \setminus \mathbb{Z}_n^\times$. Das komplettiert dann auch den Beweis. □

Ohne Kenntnis des privaten Schlüssels d besteht also die Aufgabe eines Angreifers darin, aus Kenntnis von n, e und c die Kongruenz $x^e \equiv_n c$ zu lösen. Unser e aus (3.20) setzt aber nicht die Kenntnis von n, sondern die Kenntnis von $\varphi(n) = (p-1)(p'-1) = n - p - p' + 1$ voraus. Und die ist nicht ohne.

Lemma 3.17 *Die Probleme*

1. *bestimme* $\varphi(n)$
2. *bestimme* p, p'

aus n *sind äquivalent.*

Beweis: Kennt man n und $\varphi(n)$, dann hat das quadratische Polynom

$$q(x) = x^2 - (n + 1 - \varphi(n))\,x + n = x^2 - (p + p')x + pp' = (x - p)(x - p')$$

gerade die Nullstellen p, p', die man mit einem geeigneten Lösungsverfahren[85] effizient ermitteln kann.

Kennt man umgekehrt p, p', dann ist ja $\varphi(n) = (p-1)(p'-1)$ sofort ausgerechnet. □

Bemerkung 3.18 (Berechnung von (3.21) **und** (3.22)**)** *Bei der Ver- und Entschlüsselung im RSA–Verfahren müssen Exponenten modulo* n *schnell berechnet werden, also Ausdrücke der Form* $(x^y)_n$. *Eine effiziente Berechnungsweise nennt sich* ***wiederholtes Quadrieren****.*

[85] Beispielsweise dem Newton–Verfahren, das hier *immer* konvergiert, siehe (Sauer, 2013).

1. *Hierbei wird* y *in* ***Binärform*** *als* $y = y_0 + 2y_1 + \cdots 2^n y_n$, $y_j \in \{0, 1\}$, *was zu*

$$x^y = \prod_{j=0}^{n} x^{2^j y_j} = \prod_{y_j=1} x^{2^j}$$

führt. Das kann man dann in der Schleife

$$z \leftarrow z\, x^{y_j}, \qquad x \leftarrow x^2,$$

initialisiert mit $z = x^{y_0}$ *und* $j = 1$, *sehr einfach und effizient realisieren, nämlich mit* $\log_2 y + \#\{j : y_j \neq 0\}$ *Rechenoperationen.*

2. *Einen ähnlichen Trick kann man auch zum Multiplizieren zweier Zahlen über* ***wiederholtes Verdoppeln*** *und Addition bei* $y_j = 1$ *verwenden. Interessanterweise war das die Multiplikationsmethode, die im alten Ägypten verwendet wurde, siehe*

RSA wird auch tatsächlich in der Praxis/Realität eingesetzt und zwar bei durchaus lieben Bekannten:

ssh: Das bekannte Programm zur Herstellung von Verbindungen mit entfernten Computern verwendet RSA zum Verbindungsaufbau. Dabei werden im Verzeichnis[86] `~/.ssh` in den Dateien `id_rsa` und `id_rsa.pub` ein privater und ein öffentlicher Schlüssel für Rechner und Benutzer angelegt und bei jedem neuen Verbindungsaufbau ein öffentlicher Schlüssel von dem Rechner abgefragt, mit dem man sich verbindet. Über diese Kommunikation wird dann sicher ein AES–Schlüssel vereinbart, die weitere Kommunikation läuft dann über das wesentlich schnellere AES–Protokoll.

SSL: Das ist ein ähnliches Verschlüsselungsverfahren, das für sichere Verbindungen in Browsern, also die `https://`–Seiten, verwendet wird.

PGP: Das Standardpaket zur Kryptographie[87] hat ursprünglich ebenfalls das RSA–Verfahren zum Schlüsseltausch verwendet. Inzwischen wir aber laut (Willems, 2008) ein anderes Verfahren benützt und zwar das ElGamal–Verfahren aus dem nächsten Abschnitt.

3.6 Das ElGamal–Verfahren

Während das RSA–Verfahren die Schwierigkeit der Faktorisierung von Primzahlen[88] nutzt, basiert das **ElGamal–Verfahren** auf dem diskreten Logarithmus.

Dabei wählt man wieder eine große Primzahl p und einen Erzeuger a für die zyklischen (multiplikative) Gruppe $\mathbb{Z}_p^\times = \mathbb{Z}_p \backslash \{0\}$ sowie einen Index $2 \leq b \leq p-2$, so daß also $a^b \notin \{1, a\}$ und damit auf jeden Fall verschleiert ist. Nun berechnet man noch $a' := (a^b)_p$.

[86] Zumindest bei einem **Linux**–Rechner.

[87] Und irgendwo auch der Schrecken der NSA.

[88] Natürlich müssen wir uns schon noch überlegen, ob und warum Faktorisierungen problematisch sind.

Definition 3.19 ***Öffentlicher Schlüssel*** *des ElGamal–Verfahrens ist das Tripel* $K_e := (p, a, a') = (p, a, (a^b)_p)$, ***privater Schlüssel*** *ist der Exponent* $b =: K_d$.

Die Verschlüsselung eines Blocks[89] $m \in \mathbb{Z}_p$ ist dann

$$E(K, m) = E(K_e, m) = \left((a^k)_p, (ma'^k)_p\right) =: (c_1, c_2) =: c, \tag{3.25}$$

wobei $2 \leq k \leq p-2$ ein bei der Verschlüsselung *zufällig* gewählter Wert ist. Die Entschlüsselungsfunktion ist

$$D(K, c) = D(K_d, c) = \left(c_2 (c_1^b)^{-1}\right)_p . \tag{3.26}$$

Proposition 3.20 (ElGamal) *ElGamal funktioniert, d.h. für jedes* $2 \leq k \leq p-2$ *gilt*

$$D(K, E(K, m)) = m, \qquad m \in \mathscr{M} = \mathbb{Z}_p. \tag{3.27}$$

Beweis: Wir berechnen ganz einfach

$$D(K, E(K, m)) \;=\; \left((ma'^k)_p ((a^k)_p^b)^{-1}\right)_p \equiv_p = ma'^k (a^k)^{-b} = ma^{bk}a^{-bk} = m.$$

□

Diese Decodierung ist genauso schwer wie die Bestimmung des diskreten Logarithmus, denn wenn wir das könnten, könnten wir aus c_1 den Wert von k errechnen, dann a'^k und damit $m = (c_2 a'^{-k})_p$ mit dem erweiterten euklidischen Algorithmus berechnen.

3.7 Schlüsseltausch

Die beiden letzten Verfahren bezeichnet man als **asymmetrische Verfahren**, weil sie unterschiedliche Schlüssel zur Ver- und Entschlüsselung verwenden und damit auch die Rollen von Sender und Empfänger natürlich nicht mehr symmetrisch sind. Das macht sie aber nicht nur konzeptionell sondern auch in Bezug auf die Komplexität aufwendiger und für schnelle Echtzeitanwendungen ungeeignet, bei denen man lieber symmetrische Verfahren wie **AES** verwendet. Bei diesen ist es aber notwendig, daß ein **gemeinsamer Schlüssel** ausgetauscht bzw. vereinbart wird. Das geht im „persönlichen Kontakt" einfach, ist über einen unsicheren Kanal aber etwas komplizierter. Zu diesem Zweck haben Diffie und Hellman in (Diffie & Hellman, 1976) die folgende Prozedur zum Schlüsseltausch vorgeschlagen.

Öffentlich bekannt sind eine multiplikative Gruppe G, beispielsweise $\mathbb{Z}_p$, und ein Gruppenelement $g \in G$ der Ordnung n, d.h. $g^n = 1$ und $g^k \neq 1$, $1 \leq k < n$. Jeder kennt also G, g und n. Jede der beiden Seiten[90] wählt nun einen idealerweise zufälligen Wert a bzw. b aus $\{2, \ldots, n-1\}$. Der Schlüsselaustausch

[89] Hier gilt exakt dasselbe wie bei all den anderen Verschlüsslungsverfahren vorher.

[90] Die „richtigen" Kryptographen reden dann immer von **Alice** und **Bob**, aber irgendwan wird das sowas von abgedroschen und langweilig. Vielleicht wären „Angela" und „Beer" inzwischen besser, den „Angreifer" könnte man das „NaSAweiss" nennen.

besteht nun im Austausch von g^a bzw. g^b, woraus sich beide Seiten $g^{ab} = (g^a)^b = (g^b)^a$ bestimmen können[91], und dieses g^{ab} ist dann der gemeinsame Schlüssel, den beide Seiten verwenden.

Bei dieser Methode wird also nicht der gesamte Schlüssel übertragen, sondern jeweils nur Informationen, aus denen sich zusammen mit einem geheimen Schlüssel der gemeinsame Schlüssel dann bilden lässt. Es ist laut (Willems, 2008) allerdings bis heute[92] nicht klar, ob die Sicherheit dieses Schlüsseltauschs wirklich äquivalent zur Bestimmung des diskreten Logarithmus ist.

Was aber bei dieser Methode völlig unklar ist, ist die Frage ob man den Schlüsseltausch überhaupt mit dem richtigen Partner durchführt oder mit einem „Fake", mit dem man dann seine Geheimnisse austauscht. Um das abzusichern, verwendet man meistens **Zertifikate**, die von einer vertrauenswürdigen Stelle ausgestellt worden sind. Auch die Uni Passau kann nach eingehender Prüfung (Personalausweis) Zertifikate ausstellen, die aber immer noch nicht wirklich stark sind.

3.8 Signaturen

Eine **Signatur** ist eine Art „digitale Unterschrift", die die Authentizität einer Nachricht garantiert, also sowohl den Urheber der Nachricht als auch deren Integrität sicherstellt. Anders gesagt:

1. Niemand hat die Nachricht gefälscht.

2. Niemand hat die Nachricht verändert.

Formalisieren wir das einmal; dabei können wir natürlich immer annehmen, daß die Identität des Autors einer Nachricht M einfach ein Teil derselben ist.

Definition 3.21 (Signatur–System) *Ein **Signatur–System** besteht aus einem Klartextraum $\mathscr{M}$, einem Schlüsselraum $\mathscr{K}$ und einem Signaturraum $\mathscr{S}$. Dazu kommen*

1. *eine **Signaturfunktion** $S : \mathscr{K} \times \mathscr{M} \to \mathscr{S}$ und*

2. *eine **Verifizierungsfunktion**[93] $V : \mathscr{K} \times \mathscr{S} \times \mathscr{M} \to \{0, 1\}$ mit der Eigenschaft, daß*

$$V(K, S, M) = 1 \quad \Leftrightarrow \quad S = S(K, M), \qquad S \in \mathscr{S},\ M \in \mathscr{M}. \tag{3.28}$$

Die Verifizierungsfunktion überprüft also, ob die Signatur korrekt und liefert in diesem und nur diesem Fall den Wert 1, also „korrekt" zurück, andernfalls ist das Ergebnis 0, also „falsch".

[91] Achtung: Dafür braucht man a und b! Denn $g^a g^b = g^{a+b}$ und nicht g^{ab}.

[92] Oder zumindest bis 2008.

[93] Gerne auch als **Verifikationsfunktion** bezeichnet, was aber genau genommen ein denglischer Bastard ist.

Beispiel 3.22 (RSA–Signatur) *Bei der **RSA–Signatur** verwendet man wieder das Primzahlprodukt* $n = pp'$ *und das Schlüsselpaar* d, e *aus* (3.20)*. Das liefert wieder den öffentlichen Schlüssel* $K_e = (n, e)$ *und den privaten Schlüssel* $K_d = d$*. Um eine Nachricht* $m \in \mathbb{Z}_n$ *zu signieren bildet man*

$$s := S(K, m) = (m^d)_n \tag{3.29}$$

und die Verifizierung ist der Test $m \equiv_n s^e$*, der sich nur mit öffentlichen Informationen, als mit* K_e *durchführen lässt. Der Trick ist wieder die Identität* $de = \varphi(n) + 1$*, denn damit ergibt sich für* $x, y \in \mathbb{Z}_n$

$$x^d \equiv_n y^d \quad \Rightarrow \quad \left(x^d\right)^e \equiv_n \left(y^d\right)^e \quad \Rightarrow \quad x \underbrace{x^{\varphi(n)}}_{\equiv_n 1} \equiv_n y \underbrace{y^{\varphi(n)}}_{\equiv_n 1} \quad \Rightarrow \quad x = y,$$

also

$$x^d \equiv_n y^d \quad \Leftrightarrow \quad x \equiv_n y, \qquad x, y \in \mathbb{Z}_n. \tag{3.30}$$

Als unmittelbare Konsequenz von (3.30) *ist dann aber auch*

$$m \equiv_n s^e \qquad \Leftrightarrow \qquad m^d \equiv_n s^{ed} \equiv_n s,$$

also genau dann, wenn die Signatur s *nach der Regel* (3.29) *gebildet wurde.*

Beispiel 3.23 (ElGamal–Signatur) *Die **ElGamal–Signatur** basiert wieder auf dem öffentlichen Schlüssel* (p, a, a') *und dem geheimen Wert* b*, so daß* $a' = (a^b)_p$ *ist. Nun wählt man wieder ein* zufälliges $2 \le k \le p - 2$ *mit* $\mathrm{ggT}(k, p - 1) = 1$*, genauer, man wählt* k *zufällig und testet auf Koprimalität*[94]*. Die Signaturfunktion ist dann*

$$S(K, m) = s := (s_1, s_2) := \left((a^k)_p, \left((m - bs_1)k^{-1}\right)_{p-1}\right) \tag{3.31}$$

mit dem Verifizierungstest $a'^{s_1} s_1^{s_2} \equiv_p a^m$*.*

Das witzige an der ElGamal–Signatur ist die **Zufallszahl** k, die man zwar zur Erstellung der Signatur, nicht aber deren Verfizierung benötigt und die dabei hilft, den Schlüssel b zu verschleiern. Da sich der Schlüssel aus der Kenntnis von k als

$$s_2 \equiv_{p-1} (m - bs_1)k^{-1} \qquad \Leftrightarrow \qquad b \equiv_p (ks_2 - m)s_1^{-1} \equiv_{p-1} (ks_2 - m)a^{-k}$$

bestimmen lässt, kann man ihn aus der Kenntnis von k und einer korrekt signierten Nachricht (m, u) errechnen. Deswegen ist es notwendig, daß k geheim bleibt und variiert; solange das gewährleistet ist, ist das System wegen $k = \log_a s_1$ sicher, denn die Bestimmung von k bedeutet einen diskreten Logarithmus.

Um die Korrektheit der ElGamal–Signatur nachzuweisen, müssen wir uns nur

$$a'^{s_1} s_1^{s_2} \equiv_p \left(a^b\right)^{s_1} \left(a^k\right)^{s_2} \equiv_p a^{bs_1} a^{k(m - bs_1)k^{-1}} \equiv_p a^m$$

ausrechnen, der Test gibt genau dann 1 zurück, wenn s_1 und s_2 korrekt gebildet wurden.

[94] Das ist „vornehm“ für „teilerfremd“.

Bemerkung 3.24 (Signaturen) *Bei genauem Hinsehen haben beide Signaturen ein kleines Problem: Die Unterschrift ist entweder genauso lang (RSA) oder sogar doppelt so lang (ElGamal) wie der Originaltext, die Nachrichten werden durch Signieren also massiv aufgebläht.*

Um diesem Problem entgegenzutreten und „kurze“ Unterschriften für digitale Dokumente zu finden, braucht man also eine Methode, die Signaturen zu komprimieren, aber das natürlich so, daß man immer noch vernünftig unterscheiden kann.

3.9 Hash–Funktionen

Eine **Hash–Funktion** bildet Zeichenketten beliebiger Länge,

$$x \in \mathbb{Z}_m^* := \bigcup_{n \in \mathbb{N}} \mathbb{Z}_m^n$$

über dem Alphabet $\mathbb{Z}_m$ auf kurze Zeichenketten einer *festen* Länge $n \in \mathbb{N}$ ab, ist also eine Abbildung $h : \mathbb{Z}_m^* \to \mathbb{Z}_m^n$. Beim Signieren einer Nachricht ersetzt man dann die lange Signatur s wie in Beispiel 3.22 oder Beispiel 3.23 berechnet durch den **Hashwert** $h(s)$ und vergleicht bei der Verifizierung dann auch nur die Hashwerte. Da h schon aus Mächtigkeitsgründen, $\#\mathbb{Z}_m^* = \infty > m^n = \#\mathbb{Z}_m^n$ nicht injektiv sein kann, es also Werte $x \neq x' \in \mathbb{Z}_m^*$ geben muss, so däs $h(x) = h(x')$, ist es natürlich wichtig, daß die Hashwerte eine gewisse „Trennschärfe“ haben, Funktionen wie $h = 0$ sind für Signaturen eher nicht sinnvoll.

Definition 3.25 (Kollisionsresistenz) *Eine Hash–Funktion heisst **kollisionsresistent**, wenn man zu $x \in \mathbb{Z}_m^*$ in vertretbarer Zeit kein $x' \in \mathbb{Z}_m^*$ finden kann, so daß $h(x) = h(x')$ gilt.*

Anschaulich gesprochen bedeutet Kollisionsresistenz, daß h die Daten gut „mischt“ und daß es daher nicht mglich ist, eine Hash–Funktion einfach zu fälschen, also beispielsweise eine Signatur und dazu eine Nachricht so konstruieren, daß diese ebenfalls korrekt signiert ist. Eine kollisionsresistente Funktion lässt sich wieder einmal über den diskreten Logarithmus generieren.

Definition 3.26 (Sichere Primzahl) *Eine Primzahl p heißt **sichere Primzahl**, wenn es eine Primzahl p' gibt, so daß $p = 2p' + 1$ ist.*

Satz 3.27 (Hash–Funktion) *Sei p eine sichere Primzahl, a ein **erzeugendes Element**[95] von $\mathbb{Z}_p^\times$ und $b \in \mathbb{Z}_p$. Dann ist eine Kollision der Funktion*

$$f : \mathbb{Z}_{p'}^\times \times \mathbb{Z}_{p'}^\times \to \mathbb{Z}_p, \qquad f(x_1, x_2) = \left(a^{x_1} b^{x_2}\right)_p, \tag{3.32}$$

äquivalent zur Bestimmung des diskreten Logarithmus[96] $\log_a b$.

[95] Das heisst $\mathbb{Z}_p^\times = \{a^k : k \in \mathbb{N}\}$.

[96] Zur wiederholten Wiederholung: Das ist diejenige Zahl k mit $a^k = b$.

Beweis: Seien $x = (x_1, x_2)$ und $x' = (x_1', x_2')$ eine Kollision, also $f(x) = f(x')$ bzw.

$$a^{x_1} b^{x_2} \equiv_p a^{x_1'} b^{x_2'} \qquad \Leftrightarrow \qquad a^{x_1 - x_1'} \equiv_p b^{x_2' - x_2} \equiv_p a^{\log_a b (x_2' - x_2)}.$$

Da a ein erzeugendes Element ist, ist das dazu äquivalent, daß die Exponenten modulo $\varphi(p) = p - 1$ übereinstimmen, also

$$x_1 - x_1' \equiv_{p-1} \log_a b \, (x_2' - x_2) \tag{3.33}$$

Nun sind aber $x_2, x_2' \in \mathbb{Z}_p^\times$,, wobei wir annehmen können, daß $x_2' \geq x_2$ ist, sonst vertauschen wir einfach x und x' – für die Kollision ist das ja irrelevant. Dann ist aber $x_2' - x_2 < p' = (p-1)/2$, so daß

$$\mathrm{ggT}\,(p - 1, x_2' - x_2) = \mathrm{ggT}\,(2p', x_2' - x_2) \in \{1, 2\}$$

sein muss. Hat der ggT den Wert 1, dann ist

$$\log_a b \equiv_{p-1} (x_1 - x_1')(x_2' - x_2)^{-1}$$

mit dem erweiterten euklidischen Algorithmus leicht zu berechnen, hat der ggT den Wert 2, so dividieren wir diesen aus und erhalten

$$\log_a b \equiv_{p'} (x_1 - x_1') \left(\frac{x_2' - x_2}{2} \right)^{-1}.$$

Ist umgekehrt der diskrete Logarithmus bekannt, dann wählen wir x' und x_2 beliebig und bestimmen ein kollidierendes x_1 vermittels (3.33) als Lösung von

$$x_1 \equiv_{p-1} x_1' + \log_a b \, (x_2' - x_2).$$

□

In der Praxis konstruiert man laut (Willems, 2008) Hash–Funktionen auf binäre Art und Weise. Dazu identifiziert man einfach jede Zahl $x \in \mathbb{N}$ mit den Ziffern ihrer **Binärdarstellung**

$$x = x_0 + 2x_1 + \cdots = \sum_{2^j \leq x} x_j 2^j, \qquad \text{d.h.} \qquad x = (x_0, x_1, \dots).$$

Damit ist dann $\{x \in \mathbb{N} : x < 2^n\} \simeq \mathbb{Z}_2^n$, $n \in \mathbb{N}$. Dann können wir unsere sichere Primzahl p wählen, dazu ein n mit $2^n \geq p$ und m mit $2^{m/2} < p'$ wählen und dann

$$f : \mathbb{Z}_2^m \simeq \mathbb{Z}_2^{m/2} \times \mathbb{Z}_2^{m/2} \hookrightarrow \mathbb{Z}_{p'} \times \mathbb{Z}_{p'} \to \mathbb{Z}_p \hookrightarrow \mathbb{Z}_2^n$$

als Funktion $f : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ auf binären Zeichenketten ansehen. Dabei seien m und n so gewählt, daß $r := m - n \geq 2$ ist. Bei der Konstruktion einer konkreten Hash–Funktion im Stile des **SHA–Standard** wollen wir nun Bitfolgen beliebiger aber endlicher Länge $x \in \mathbb{Z}_2^*$ hashen. Dazu gehen wir folgendermaßen vor:

1. Fülle x von links so mit Nullen auf, daß die Länge von x ein Vielfaches von r ist und hänge danach am Ende von x nochmals r Nullen an. Dann ist $x \in \mathbb{Z}_2^{kr}$ für ein $k \in \mathbb{N}$.

2. Nun schreibt man die Länge der Bitfolge x ebenfalls als *binäre* Zahl und füllt diese von links mit einer *minimalen* Anzahl von Nullen auf, so daß die Länge dieser Folge durch $r-1$ teilbar ist, wir betten also die Länge in $\mathbb{Z}_2^{\ell(r-1)}$ ein. Jedes der ℓ Segmente der Länge $r-1$ dieser Folge bekommt nun noch eine 1 vorangstellt, was letztendlich eine Folge $y \in \mathbb{Z}_2^{\ell r}$ liefert.

3. Die **Konkatenation** $xy \in \mathbb{Z}_2^{(k+\ell)r}$ wird nun als eine Folge

$$z = (z_j \in \mathbb{Z}_2^r \,:\, j \in \mathbb{Z}_t) \in (\mathbb{Z}_2^r)^t\,, \qquad t := k+\ell, \tag{3.34}$$

interpretiert.

Das ist zugegebenermaßen ein klein wenig verwirrend, weswegen wir uns die Sache mal an einem Beispiel anschauen wollen. Man sollte beachten, daß die konkreten Werte von m und n hier noch *keine* Rolle spielen, sondern nur die Differenz r.

Beispiel 3.28 *Sei $r = 3$ und $x = 1011110$ eine Bitfolge der Länge 7.*

1. *Auffüllen von links auf 9 Zeichen und Anhängen von $r = 3$ Nullen*

$$x \mapsto 001011110000 = 001\,011\,110\,000.$$

2. *Die Länge $7 = 111$ wird auf ein Vielfaches von $r-1 = 2$ aufgefüllt,*

$$\ell(x) \mapsto 0111 = 01\,11$$

und jedem der Segmente eine 1 vorangestellt:

$$\ell(x) \mapsto \underbrace{101}\,\underbrace{111} = 101\,111$$

3. *Hintereinanderhängen der beiden Folgen:*

$$x \mapsto z = \underbrace{001\,011\,110\,000}\,\underbrace{101\,111}\,.$$

Nun interpretieren wir unsere Hashfunktion $f : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ als Funktion $\mathbb{Z}_2^n \times \mathbb{Z}_2^r \to \mathbb{Z}_n$ und iterieren diese ausgehend von $h_0 := 0 \in \mathbb{Z}_2^n$ mit Notation von (3.34) folgendermaßen:

$$h_{j+1} = f\,(h_j, z_j)\,, \quad j \in \mathbb{Z}_t, \qquad h(x) := h_t. \tag{3.35}$$

Satz 3.29 (Kollisionsresistenz) *Die Hash–Funktion h aus (3.35) ist kollisionsresistent, wenn sich der diskrete Logarithmus in $\mathbb{Z}_p^\times$ nicht effizient berechnen lässt.*

Beweis: Seien $x \neq x'$ zwei Bitfolgen mit $h(x) = h(x')$ und zugehörigen z_j, z'_j sowie $t \leq t'$ – letzteres können wir aus Symmetriegründen immer annehmen.

Nun ist

$$f(h_{t-1}, z_{t-1}) = h_t = f(x) = f(x') = h'_{t'} = f(h'_{t'-1}, z'_{t'-1})$$

und wäre $h_{t-1} \neq h'_{t'-1}$ oder $z_{t-1} \neq z'_{t'-1}$, so hätten wir eine Kollision von f gefunden und damit die Fähigkeit, den diskreten Logarithmus zu berechnen. Andernfalls ist $h_{t-1} = h'_{t'-1}$ und damit

$$f(h_{t-2}, z_{t-2}) = h_{t-1} = h'_{t'-1} = f(h'_{t'-2}, z'_{t'-2}),$$

was sich induktiv entweder auf das Auffinden einer Kollision von f oder

$$h_{t-k} = h'_{t'-k}, \qquad z_{t-k} = z'_{t'-k}, \qquad k \leq t, \tag{3.36}$$

erweitern lässt. Das bedeutet aber entweder, daß dann doch $t = t'$ und $z = z'$, also auch $x = x'$ wäre, oder aber den Widerspruch

$$0 = h_0 = f\left(h_{t'-t-1}, z_{t'-t-1}\right).$$

□

Übung 3.3 Warum ist $f(h, z) = 0$ unmöglich? ◇

Das hier vorgestellte Verfahren ist nach (Willems, 2008) in ähnlicher Form Bestandteil des **SHA-1**–Standard, der sich allerdings 2005 aufgrund seiner zu kurzen Schlüssellänge wegen des iterativen Ansatzes als schwach herausgestellt hat und seitdem durch **SHA-256** ersetzt werden soll.

God may not play dice with the universe, but something strange is going on with the prime numbers.

P. Erdös

4 Spaß mit Primzahlen

Nachdem wir nun einige kryptographische Verfahren kennengelernt haben, die auf der Schwierigkeit beruhen, Primfaktorzerlegungen zu berechnen, wird es Zeit, sich nun einmal ernsthafter mit Primzahlen zu befassen. Das Augenmerk dabei liegt auf Verfahren, die

1. testen, ob eine gewisse Zahl eine Primzahl ist,
2. die Primfaktorzerlegung einer Zahl berechnen,
3. Primzahlen einer gewissen Größe berechnen.

Das sind natürlich unterschiedlich schwierige Aufgaben: Wenn man eine Zahl faktorisieren kann, dann weiß man natürlich auch, ob sie eine Primzahl ist oder nicht.

Unabhängig vom Nützlichkeitsaspekt im kryptographischen Kontext waren und sind Primzahlen aber auch einfach aufgrund ihrer Eigenschaften interessante mathematische Studienobjekte, die immer schon einiges an Aufmerksamkeit auf sich gezogen haben.

4.1 Einfache Eigenschaften

Beginnen wir mal mit der formalen Definition.

Definition 4.1

1. *Eine Zahl* $p \in \mathbb{Z}$ *heißt* ***Primzahl****, wenn sie nur von* 1 *und* p *geteilt wird, bzw. wenn* $\mathrm{ggT}(p, a) \in \{1, p\}$, $a \in \mathbb{Z}$, *ist. Die Menge aller Primzahlen bezeichnen mir mit* $\mathbb{P} \subset \mathbb{Z}$.

2. *Die* ***Primfaktorzerlegung*** *einer Zahl* $a \in \mathbb{Z}$ *ist das Produkt*

$$a = (-1)^{\varepsilon} \prod_{j=1}^{n} p_j^{e_j}, \qquad p_j \in \mathbb{P},\ e_j \in \mathbb{N},\ \epsilon \in \{0, 1\}. \tag{4.1}$$

3. *Man numeriert die Primzahlen gelegentlich auch als* p_j, $j \in \mathbb{N}$, *durch, mit* $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, *usw.*

Die Primzahlen sind ein unerschöpfliches Reservoir, wie bereits die alten Griechen wussten.

Lemma 4.2 (Euklid)

1. *Jede Zahl $a \in \mathbb{Z}$ hat eine bis auf Vorzeichen*[97] *Primfaktorzerlegung.*

2. *Es gibt unendlich viele Primzahlen.*

Beweis: Ist $a \in \mathbb{Z}$ bereits eine Primzahl p, so ist $a = p$ bereits die gewünschte Zerlegung. Andernfalls hat a einen echten Teiler a_1, also $a = a_1\, a_2$, $1 < |a_1|, |a_2| < |a|$, auf die man dann per Induktion die Behauptung 1) anwenden kann, womit 1) auch schon bewiesen ist.

Angenommen, es gäbe nur endliche viele $p_1, \ldots, p_n$, dann gilt für $q = p_1 \cdots p_n + 1$ offensichtlich $(q)_{p_j} = 1, j = 1, \ldots, n$, also ist entweder q eine Primzahl oder zerfällt in Primfaktoren, die mit $p_1, \ldots, p_n$ nichts zu tun haben. Auf alle Fälle gibt es aber mehr Primzahlen als $p_1, \ldots, p_n$ und da n beliebig war, muss $\#\mathbb{P} = \infty$ sein. □

Die Standardmethode zur Aufzählung aller Primzahlen $\leq N$ für $N \in \mathbb{N}$ ist der **Sieb des Erathostenes**, der wie folgt funktioniert:

1. Beginne mit $P = \emptyset$ und $L = \{2, \ldots, N\}$.

2. Entferne alle Vielfachen des *kleinsten* Elements $\ell = \min L$ aus L und füge ℓ zu P hinzu.
$$L \leftarrow L \setminus \ell\,\mathbb{N}, \qquad P \leftarrow P \cup \{\ell\}.$$

3. Das wird wiederholt bis $L = \emptyset$ ist.

Dann enthält P am Ende $\mathbb{P} \cap \{2, \ldots, N\}$, was man einfach aus der Invarianten des Verfahrens ableitet: P enthält nur Primzahlen und kein Element von L wird von einem Element von P geteilt. Ausserdem ist $\{1, \ldots, N\} = P\,\mathbb{N} \cup L$.

Eine deutlich spannendere Frage ist dann schon, wie groß eigentlich die Primzahldichte ist, also wieviele Primzahlen man beispielsweise so in $\{1, \ldots, N\}$ erwarten kann. Dazu betrachtet man die Funktion $\pi : \mathbb{R} \to \mathbb{N}$, definiert als

$$\pi(x) := \#\{p \in \mathbb{P} \;:\; p \leq x\}, \tag{4.2}$$

aus der sich die **Primzahldichte** als $\pi_d(x) := x^{-1}\pi(x)$ ergibt. Diese Dichte kann man gut beschreiben, sie ist asymptotisch $1/\log x$.

Satz 4.3 (Primzahlsatz)

$$\lim_{x\to\infty} \pi_d(x)\, \log x = 1, \tag{4.3}$$

wobei hier „$\log = \log_e$“ für den natürlichen Logarithmus steht[98].

[97] Das heisst, bis auf *Einheiten* in $\mathbb{Z}$.

[98] Andernfalls bekommt man eine andere Konstante.

Diesen Satz wollen wir nicht beweisen, und der Beweis scheint auch durchaus aufwendig zu sein. Die folgende, etwas schwächere, Form des Primzahlsatzes findet sich (Andrews, 1971), wo sie auch relativ elementar bewiesen wird.

Satz 4.4 (Tscheyscheff) *Für* $x \geq 8$ *ist*

$$\frac{\log 2}{4} \frac{x}{\log x} < \pi(x) < 30 \log 2 \frac{x}{\log x}. \tag{4.4}$$

Auch hier gibt es keinen Beweis, dafür aber ein anderes Resultat, das auf Euler zurückgeht und eine Verschärfung der Aussage von Euklid in Lemma 4.2 darstellt und darauf basiert, daß die **harmonische Reihe** $\sum \frac{1}{n}$ divergiert.

Satz 4.5 (Euler)

$$\sum_{p \in \mathbb{P}} \frac{1}{p} = \infty. \tag{4.5}$$

Beweis: Wir betrachten die *konvergente* Reihe

$$\sum_{k=1}^{\infty} \frac{1}{k^s}, \qquad s > 1,$$

und eine Auflistung $p_1 < p_2 < \cdots$ von p. Wegen der eindeutigen Primfaktorzerlegung besteht eine $1-1$–Beziehung

$$k \leftrightarrow p_1^{k_1} p_2^{k_2} \cdots = \prod_{j=1}^{\infty} p_j^{k_j},$$

wobei fast alle k_j den Wert 0 haben[99]. Damit ist

$$C_s := \sum_{k=1}^{\infty} \frac{1}{k^s} = \sum_{k_1=0}^{\infty} \sum_{k_2=0}^{\infty} \cdots \left(\prod_{j=1}^{\infty} p_j^{k_j} \right)^{-s} = \prod_{j=1}^{\infty} \sum_{k_1=0}^{\infty} \frac{1}{p_j^{k_j s}} = \prod_{j=1}^{\infty} \left(1 - p_j^{-s}\right),$$

und da $C_s \to \infty$ für $s \to 1$ divergiert das unendliche Produkt der $1 - 1/p_j$ und damit, nach Lemma 4.7, das wir gleich noch beweisen werden, auch die Reihe aus (4.5). □

Übung 4.1 Zeigen Sie:

$$\sum_{k=1}^{\infty} \frac{1}{k} = \infty, \qquad \sum_{k=1}^{\infty} \frac{1}{k(k+1)} = 1.$$

♢

Bemerkung 4.6 (Primzahldichte)

[99] Was uns aber nicht interessieren soll.

1. *Satz 4.5 sagt uns nicht nur, daß es unendlich viele Primzahlen geben muss, denn sonst würde die Reihe ja nicht divergieren, sondern daß die Primzahlen mit einer gewissen Häufigkeit auftreten müssen und zwar häufiger als die Zahlen k^s, $s > 1$.*

2. *Insbesondere sind Primzahlen viel häufiger als Quadratzahlen.*

3. *Erinnert man sich an den Divergenzbeweis für die harmonische Reihe, so kann man sich sogar ungefähr klarmachen, warum Satz 4.3 gelten muss. Bei dem Beweis zerlegt man ja*

$$\sum_{k=1}^{\infty} k^{-1} = \sum_{j=1}^{\infty} \sum_{k=a^{j-1}+1}^{a^j} j^{-1}, \qquad a \geq 2,$$

und zeigt, daß sich die innere Summe durch eine von j unabängige Konstante nach unten[100] *abschätzen lässt. Wenn nun auch die Primzahlreihe divergiert, dann sollte auch*

$$\#\left(\mathbb{P} \cap \{a^{j-1}+1, \ldots, a^j\}\right) = c_j, \qquad j \in \mathbb{N},$$

zumindest keine Nullfolge sein und daraus lässt sich auf eine mindestens logarithmische Dichte schliessen[101]*. Das ist aber aller reine Heuristik hier, der „richtige" Beweis ist schon ein bisschen aufwendiger, aber eben auch Stoff für eine Vorlesung zur Zahlentheorie.*

Ein **unendliches Produkt** muss man mit ein wenig Vorsicht behandeln, da es sich normalerweise in der Menge $\mathbb{R}_+ = \{x \in \mathbb{R} : x > 0\}$ abspielt, die eine **mutliplikative Gruppe** mit neutralem Element 1 ist, aber eben auch eine offene und, noch viel schlimmer, eine nicht abgeschlossene Menge. Der Randpunkt 0 gehört ebenso wenig dazu wie der andere Randpunkt ∞. Daher spricht man beim Grenzwert 0 eines unendlichen Produkts genauso von **Divergenz**, als wenn der Grenzwert ∞ wäre. Eigentlich spielt die 0 in $\mathbb{R}_+$ dieselbe Rolle wir $-\infty$ im additiven $\mathbb{R}$. Fehlt noch das Lemma zur Konvergenz uendlicher Produkte.

Lemma 4.7 *Für $a_j \in [0, 1)$, $j \in \mathbb{N}$, hat das unendliche Produkt*

$$\prod_{j=0}^{\infty} (1 - a_j)$$

genau dann einen positiven Grenzwert[102]*, wenn die unendliche Reihe*

$$\sum_{j=0}^{\infty} a_j$$

konvergiert.

[100] Und auch nach oben.

[101] Als Lagrange und Euler die ersten Resultate zur logarithmischen Primzahldichte veröffentlichten, meinte Gauß nur, er hätte das alles schon lange gewusst.

[102] Also einen Grenzwert im strengen Sinne.

Beweis: Da die endlichen Teilprodukte $(1-a_1)\cdots(1-a_n)$, $n \in \mathbb{N}$, eine monoton fallende Folge von positiven Zahlen bilden, ist es klar, daß der Grenzwert

$$0 \le \lambda = \prod_{j=0}^{\infty}(1-a_j) = \lim_{n\to\infty}\prod_{j=0}^{n}(1-a_j)$$

in $\mathbb{R}$ existiert – die einzige Frage ist, ob er positiv oder Null ist. Außerdem sieht man sofort, daß $\lambda = 0$ ist, wenn a_j keine Nullfolge ist. Also müssen wir uns beim Beweis von Lemma 4.7 nur für Nullfolgen wirklich „anstrengen".

Die einfache Idee für den Beweis ist die Abschätzung

$$e^{-2x} \le 1-x \le e^{-x}, \qquad 0 \le x \le \frac{1}{2}\log 2, \tag{4.6}$$

denn an $x = 0$ haben alle drei Ausdrücke den Wert 1 und für die Ableitungen gilt

$$-2e^{-2x} \le -1 \le -e^{-x}, \qquad 0 \le x \le \frac{1}{2}\log 2.$$

Ist also nun a_j eine Nullfolge, dann ist $a_j < \frac{1}{2}\log 2$ für $j \ge n_0$ und wir haben, daß

$$\prod_{j=n_0}^{\infty}(1-a_j) \ge \prod_{j=n_0} e^{-2a_j} = \exp\left(-2\sum_{j=n_0}^{\infty} a_j\right) \tag{4.7}$$

sowie

$$\prod_{j=n_0}^{\infty}(1-a_j) \le \prod_{j=n_0} e^{-a_j} = \exp\left(-\sum_{j=n_0}^{\infty} a_j\right). \tag{4.8}$$

Konvergiert nun die Reihe, so konvergiert auch die Teilreihe, die bei n_0 beginnt, sagen wir gegen a und (4.7) liefert uns, daß

$$\lambda \ge e^{a}\prod_{j=0}^{n_0-1}(1-a_j) > 0,$$

divergiert die Reihe hingegen, so liefert uns (4.8), daß

$$\lambda \le e^{-\infty}\prod_{j=0}^{n_0-1}(1-a_j) = 0,$$

was genau die Behauptung war. □

Ein anderer Traum der Mathematiker wäre eine **Primzahlformel**, also eine Formel, die, wenn sie schon nicht die Primzahlen direkt aufzählt, so doch immer Primzahlen liefert. Daran hat sich schon Fermat[103] versucht.

[103] Pierre de Fermat, 1601–1665, hauptberuflich Richter, wohl **der** herausragende Amateurmathematiker. Ein empfehlenswertes Buch über die große Fermatsche Vermutung und deren Beweis ist übrigens (Singh, 2000a).

Definition 4.8 *Die* n*–te* ***Fermat–Zahl*** *ist die Zahl* $f_n := 2^{2^n} + 1$.

Fermat vermutete, daß alle so generierten Zahlen Primzahlen seien, wohl aus der Kenntnis der ersten Spezialfälle

$$\begin{aligned} f_0 &= 2^{2^0} + 1 = 2^1 + 1 = 2 + 1 = 3, \\ f_1 &= 2^{2^1} + 1 = 2^2 + 1 = 4 + 1 = 5, \\ f_2 &= 2^{2^2} + 1 = 2^4 + 1 = 16 + 1 = 17, \\ f_3 &= 2^{2^3} + 1 = 2^8 + 1 = 256 + 1 = 257, \\ f_4 &= 2^{2^4} + 1 = 2^{16} + 1 = 65536 + 1 = 65537. \end{aligned}$$

und der Tatsache, daß die Zahlen dann unangenehm groß werden und es viel zu rechnen gibt. In der Tat dauerte es fast 100 Jahre bis schließlich Euler die Faktorisierung

$$f_5 = 2^{2^5} + 1 = 2^{32} + 1 = 4294967297 = 641 \cdot 6700417$$

angeben und so Fermat widerlegen konnte.

Jede Fermat–Zahl ist übrigens eine spezielle **Mersenne–Zahl**[104], das sind Zahlen der Form $2^n - 1$ und diese stellen auch die reichste Fundgrube für Primzahlen dar. Laut (Gathen & Gerhard, 1999) sind die größten bekannten Primzahlen allesamt Mersenne–Zahlen.

4.2 Probabilistische Primzahltests

Wir kommen also zu der interessanten Frage, wie wir *testen* können, ob eine gegebene Zahl $n \in \mathbb{N}$ eine Primzahl ist oder nicht. Der einfachste und naive Ansatz wäre die Erathostenes–Methode, durch alle Primzahlen $\leq \sqrt{n}$ zu dividieren und wenn das schiefgeht, dann ist n eine Primzahl.

Auch hier wird uns wieder die **Euler–Funktion** zur Hilfe kommen, deren besondere Rolle wir ja schon kennengelernt haben. Zur Erinnerung: Ist n eine Primzahl, dann ist $\varphi(n) = n - 1$, ist $n = p^e$ eine **Primzahlpotenz**, dann gilt $\varphi(n) = p^{e-1}(p-1)$ und für die allgemeine Primfaktorzerlegung aus (4.1) erhalten wir

$$\varphi(n) = \prod_{j=1}^{k} p_j^{e_j - 1}(p_j - 1).$$

Definition 4.9 *Für eine* ***koprime Zahl*** $a \in \mathbb{Z}_n$*, also ein* a *mit* $\mathrm{ggT}(a, n) = 1$*, definieren wir die* ***Ordnung*** $\omega(a)$ *von* a *als*

$$\omega(a) := \min\left\{k \geq 1 \ : \ a^k \equiv_n 1\right\}. \tag{4.9}$$

a *wird dann auch als* ***primitive Einheitswurzel*** *der Ordnung* $\omega(a)$ *von in* $\mathbb{Z}_n$ *bezeichnet.*

[104] Marin Mersenne, 1588–1648, französischer Mönch mit wichtigen Beiträgen zur Zahlentheorie, insbesondere Primzahlen. Darüber hinaus geht aber auch ein fundamentales Werk zur Musiktheorie, die „Harmonie Universelle“ (Mersenne, 2003) auf ihn zurück, die auch eine akribische Zusammenstellung der Musikinstrumente seiner Zeit enthält, siehe (Köhler, 1987).

Jetzt noch schnell zwei wichtige Fakten zur Ordnung von Elementen.

Lemma 4.10 *Für $n \in \mathbb{N}$ gilt:*

1. *Ist $\mathrm{ggT}(a,n) = 1$ und $a^k \equiv_n 1$, dann gilt $\omega(a)|k$.*
2. *Ist $n = p^e$ für eine Primzahl p und $e \geq 2$, dann ist $\omega(1+p^{e-1}) = p$.*

Beweis: 1) haben wir ganz ähnlich schon im Abschnitt 3.3 gesehen. Da $\omega(a) \leq k$, können wir die Division mit Rest $a = q\omega(a) + r$, $r < \omega(a)$ verwenden und erhalten

$$1 \equiv_n a^k = a^{q\omega(a)+r} = (\underbrace{a^{\omega(a)}}_{\equiv_n 1})^q + a^r \equiv_n a^r,$$

was nur dann klappt, wenn $r = 0$ ist.

Für 2) berechnen wir

$$\begin{aligned}(1+p^{e-1})^p &= \sum_{j=0}^{p} \binom{p}{j} p^{j(e-1)} = 1 + p\,p^{e-1} + \sum_{j=2}^{p} \binom{p}{j} p^{j(e-1)} \\ &= 1 + p^e + p^e \sum_{j=2}^{p} \binom{p}{j} p^{(j-1)(e-1)-1} \equiv_{p^e} 1,\end{aligned}$$

da für $e, j \geq$ natürlich auch $(j-1)(e-1) \geq 1$, also $(j-1)(e-1)-1 \geq 0$ ist. □

Ein erster einfacher Primalitätstest basiert auf dem kleinen Fermat und entscheidet zwischen „reduzibel“ und „ich weiss nicht“. Daß die zu untersuchende Zahl ungerade, also $\in 2\mathbb{N}+1$, sollte klar sein, denn die Menge der geraden Primzahlen ist von Haus aus recht überschaubar.

Algorithmus 4.11 (Fermat–Test)
Gegeben: *$n \in 2\mathbb{N}+1$.*

1. *Wähle ein zufälliges $a \in \{3, \ldots, n-2\}$.*
2. *Berechne*[105] *$b = (a^{n-1})_n$.*
3. *Ist $b \not\equiv_n 1$, dann ist n **keine** Primzahl, ansonsten gebe „weiß nicht“ aus.*

Ergenis: *„zerlegbar“ oder „weiß nicht“.*

Die Korrektheit des Verfahrens basiert auf dem kleinen Fermat, Korollar 2.42, denn wäre n prim, dann wäre ja $1 \equiv_n a^{\varphi(n)} = a^{n-1}$. Dieses a wäre dann ein **Zeuge** dafür, daß n eine zusammengestzte Zahl ist. Unglücklicher ist allerdings der andere Fall. Um den zu verstehen, betrachten wir die Menge

$$L_n := \left\{ a \in \mathbb{Z}_n^\times \,:\, a^{n-1} \equiv_n 1 \right\},$$

[105] Idealerweise wieder durch **wiederholtes Quadrieren**, siehe Bemerkung 3.18.

die eine multiplikative **Untergruppe** von $\mathbb{Z}_n^\times$ darstellt. Da die Kardinalität jeder Gruppe eine Vielfaches jeder ihrer Untergruppen ist[106] ist, gilt

$$\#L_n \,|\, \#\mathbb{Z}_n^\times \qquad \Rightarrow \qquad \#L_n \le \frac{1}{2}\#\mathbb{Z}_n^\times = \frac{1}{2}\,\varphi(n) \le \frac{n-1}{2} = \#\mathbb{Z}_n \tag{4.10}$$

Ist also das a aus Algorithmus 4.11 aus $\mathbb{Z}_n^\times \setminus L_n$ gewählt und ist n keine Primzahl dann erhalten wir auf jeden Fall „zerlegbar", und das nach (4.10) mit einer Wahrscheinlichkeit von

$$p(a \in \mathbb{Z}_n^\times \setminus L_n) = \frac{\#\mathbb{Z}_n \setminus L_n}{\#\mathbb{Z}_n} = 1 - \frac{\#L_n}{n-1} \ge 1 - \frac{1}{2} = \frac{1}{2}.$$

Und das macht aus dem prinzipiell „billigen" Fermat–Test ein durchaus starkes probabilistisches Verfahren.

Satz 4.12 (Fermat–Test) *Ist $n \in \mathbb{N}$ keine Primzahl, so ist die Wahrscheinlichkeit, daß der Fermat–Test dies in k Versuchen nicht erkennt, höchstens 2^{-k}.*

Beweis: Das ist jetzt einfach: Ist n keine Primzahl, dann ist in jedem Versuch, die Wahrscheinlichkeit für „zerlegbar" $\ge \frac{1}{2}$, und damit die Wahrscheinlichkeit für „weiß nicht" $\le \frac{1}{2}$. Damit kommt aber wegen der Unabhängigkeit der Versuche[107] auf eine Wahrscheinlickeit $\le \left(\frac{1}{2}\right)^k$ für k–faches „weiß nicht". □

Bemerkung 4.13 *Im Normalfall wird der Fermat–Test sogar noch viel besser funktionieren, denn die Wahrscheinlichkeit für ein sicheres „zerlegbar" erfüllt ja bei genauerem Hinsehen*

$$p(a \in \mathbb{Z}_n^\times \setminus L_n) \le 1 - \frac{\varphi(n)}{2(n-1)},$$

und je „zerlegbarer" die Zahl n ist, desto kleiner wird dieser Faktor, genauer gilt für $n = p_1^{e_1} \cdots p_k^{e_k}$

$$\frac{\varphi(n)}{2(n-1)} = \frac{n}{2(n-1)} \prod_{j=1}^{k} \left(1 - \frac{1}{p_j}\right), \tag{4.11}$$

so daß insbesondere kleine Primfaktoren die Wahrscheinlichkeit für „weiß nicht" deutlich verringern.

Übung 4.2 Beweisen Sie (4.11). ◊

Übung 4.3 Implementieren Sie den Fermat–Test in `Matlab` bzw. `Octave`. ◊

Besonders schlecht für den Fermat–Test sind Zahlen, bei denen der Test sehr oft „weiß nicht" liefert, ohne daß sie aber Primzahlen sind.

[106] Das ist der **Satz von Lagrange**.

[107] Was übrigens einschließt, daß man auch mal versehentlich dasselbe a mehrfach testet, aber was soll's? Sich die Versuche zu merken kostet nur Speicher und Zeit.

Definition 4.14 (Carmichael–Zahl) *Eine Zahl* $n \in \mathbb{N}$ *heißt* ***Charmichael–Zahl*** *wenn*

$$\mathrm{ggT}(a, n) = 1 \quad \Rightarrow \quad a^{n-1} \equiv_n 1. \tag{4.12}$$

Satz 4.15 *Eine ungerade*[108] *Zahl* $n \in \mathbb{N}$ *mit Primfaktorzerlegung* $n = p_1^{e_1} \cdots p_k^{e_k}$ *ist genau dann eine Charmichael–Zahl, wenn*

$$k \geq 3, \qquad e_j = 1, \qquad (p_j - 1) \,|\, (n-1). \tag{4.13}$$

Beweis: Für „⇒“ greifen wir uns einen Primfaktor heraus und schreiben die Zahl n als $n = p^e n'$ mit $\mathrm{ggT}(p, n') = 1$. Die multiplikative Gruppe $\mathbb{Z}_{p^e}^\times$ ist **zyklisch**[109], d.h., es gibt ein $a \in \mathbb{Z}_{p^e}^\times$ mit $\mathbb{Z}_{p^e}^\times = \left\{ a^k \,:\, k \in \mathbb{Z}_{p^{e-1}(p-1)} \right\}$, siehe (2.32). Nach dem chinesischen Restsatz gibt es dann ein $b \in \mathbb{Z}$ mit

$$b \equiv_{p^e} a \qquad \text{und} \qquad b \equiv_{n'} 1$$

und damit $\mathrm{ggT}(b, n) = 1$. Da n nach Annahme eine Charmichael–Zahl ist, ist $b^{n-1} \equiv_n 1$ und damit auch $b^{n-1} \equiv_{p^e} 1$, auch diesmal wieder nach dem chinesischen Restsatz, also $a^{n-1} \equiv_{p^e} b^{n-1} \equiv_{p^e} 1$. Damit gilt aber wieder einmal wie in Satz 2.41, daß

$$p^{e-1}(p-1) = \omega(a) \,|\, (n-1)$$

also $e = 1$ da $\mathrm{ggT}(n, p) = p$, also $\mathrm{ggT}(n-1, p) = 1$, und damit $p - 1 | n - 1$. Bleibt noch zu zeigen, daß wir mindestens drei Primzahlen brauchen. Sei also $n = p_1 p_2$ mit $p_1 > p_2$. Dann ist

$$n - 1 = p_1 p_2 - 1 = (p_1 - 1)p_2 + p_2 - 1 \equiv_{p_1 - 1} p_2 - 1.$$

Da $p_1 > p_2$ ist kann aber $p_1 - 1$ kein Teiler von $p_2 - 1$ sein und damit auch keiner von $n - 1$.

Für „⇐“ setzen wir $d_j = (n-1)/(p_j - 1)$ und suchen uns ein $c \in \mathbb{Z}$ mit $\mathrm{ggT}(c, n) = 1$. Dann ist für $j = 1, \ldots, k$

$$c^{n-1} = c^{d_j(p_j - 1)} = \left(c^{p_j - 1}\right)^{d_j} \equiv_{p_j} 1,$$

also, nach dem chinesischen Restsatz, $c^{n-1} \equiv_n 1$. □

Bemerkung 4.16 (Charmichael–Zahlen) *Es gibt unendlich viele Charmichael–Zahlen*[110] *und die kleinste unter ihnen ist* 561*. Damit gibt es leider auch unendlich viele Zahlen, für die der Fermat–Test ein echtes „weiß nicht“ ausgibt.*

Übung 4.4 Zeigen Sie, daß 561 eine Charmichael–Zahl ist. ◊

Für einen leistungsfähigeren Test, der natürlich dann auch etwas aufwendiger sein wird, brauchen wir, was auch niemanden mehr verwundern sollte, etwas mehr Theorie.

[108] Die geraden Zahlen sind im Moment sowieso eher uninteressant.

[109] Das werden wir jetzt ausnahmsweise nicht auch noch beweisen, das ist etwas für Algebra und Zahlentheorie.

[110] Das wurde bereits 1910 von Charmichael vermutet, aber erst 1994 in (Alford *et al.*, 1994) bewiesen.

Lemma 4.17 *Sei* $p =: 2^s q + 1$, $q \in 2\mathbb{N}+1$, *eine ungerade Primzahl und* $a \in \mathbb{N}$ *mit* $\mathrm{ggT}(a,p) = 1$. *Dann ist entweder*

$$a^q \equiv_p 1 \qquad \textit{oder} \qquad a^{2^r q} \equiv_p -1 \tag{4.14}$$

für ein $r \in \mathbb{Z}_s$.

Beweis: Mal wieder kleiner Fermat:

$$a^{p-1} \equiv_p 1 \qquad \Rightarrow \qquad p \mid a^{p-1} - 1 = \left(a^{(p-1)/2} + 1\right)\left(a^{(p-1)/2} - 1\right)$$

liefert

$$\left(a^{(p-1)/2}\right)_p \in \{-1, 1\}.$$

Ist $a^{(p-1)/2} \equiv_p -1$, dann ist die rechte Alternative von (4.14) mit $r = s-1$ erfüllt, wenn nicht, dann wiederholen wir das Argument von oben mit dem Ergebnis $\left(a^{(p-1)/4}\right)_p \in \{-1, 1\}$. Das macht man nun so lange, bis entweder ein Divisionsrest den Wert -1 hat oder man aber bei der linken Alternative $a^q \equiv_p 1$ angelangt ist. □

Ist nun p aus Lemma 4.17 *keine* Primzahl, dann können die beiden Bedingungen aus (4.14) auch nicht erfüllt sein, was die folgende Definition motiviert.

Definition 4.18 *Sei* $n = 2^s q + 1$, $q \in 2\mathbb{Z}+1$. *Eine Zahl* $k \in \{2, \ldots, n-1\}$ *heißt* ***Zeuge*** *für die Zerlegbarkeit von* n, *falls*

$$a^q \not\equiv_n 1 \qquad \textit{und} \qquad a^{2^r q} \not\equiv_n -1, \quad 0 \le r < s, \tag{4.15}$$

erfüllt ist.

Das führt dann auch schon zum **Miller–Rabin–Test**. Hierbei wählt man zu einer *ungeraden*[111] Zahl $n \in \mathbb{N}$ zufällig Zahlen $a \in \{2, \ldots, n-1\}$ und testet zuerst einmal, ob $\mathrm{ggT}(a,n) = 1$, denn wenn das nicht erfüllt ist, dann ist n definitiv keine Primzahl. Ansonsten teste man, ob a ein Zeuge für die Zerlegbarkeit ist und gibt „vielleicht prim“ aus, wenn man nach k Versuchen keinen Zeugen gefunden hat.

Man kann allerding zeigen, daß für zusammengesetzte Zahlen

$$\#\{a \in \{2, \ldots, n-1\} \,:\, a \text{ ist Zeuge}\} \ge \frac{3\varphi(n)}{4}, \qquad n \ge 9,$$

ist, und da $\#\{a \,:\, \mathrm{ggT}(a,n) = 1\} = \varphi(n)$, ist also die Wahrscheinlichkeit, einen Zeugen zu erwischen $\ge \frac{3}{4}$. Damit ist die Wahrscheinlichkeit, fälschlicherweise „vielleicht prim“ auszugeben, dann aber $\le 4^{-n}$.

Definition 4.19 *Das* ***Legendre–Symbol*** $\left(\frac{a}{p}\right)$ *einer Zahl* $a \in \mathbb{N}$ *zu einer ungeraden Primzahl* p *ist definiert als*

$$\left(\frac{a}{p}\right) := \begin{cases} 0, & p \mid a, \\ 1, & a \equiv_p b^2, \\ -1, & \textit{sonst}, \end{cases} \tag{4.16}$$

bestimmt also, a *modulo* p *eine Quadratzahl ist oder nicht.*

[111] Schon sowas wie eine notwendige Voraussetzung für eine Primzahl!

Lemma 4.20 (Euler) *Ist* p *eine Primzahl und* $a \in \mathbb{Z} \setminus p\mathbb{Z}$*, dann gilt*

$$a^{(p-1)/2} \equiv_p \left(\frac{a}{p}\right). \tag{4.17}$$

Beweis: Ist $\left(\frac{a}{p}\right) = 1$, also $a \equiv_p b^2$ eine Quadratzahl, dann ist

$$a^{(p-1)/2} = b^{p-1} = b^{\varphi(p)} \equiv_p 1$$

nach dem kleinen Fermat und für den Fall $\left(\frac{a}{p}\right) = 1$ stimmt das Lemma schon einmal. Für $\left(\frac{a}{p}\right) = 0$ ist die Sache ziemlich trivial und im Fall $\left(\frac{a}{p}\right) = -1$ wählen wir einen Erzeuger g von $\mathbb{Z}_p^\times$ und erhalten, daß $a = g^{2k+1}$ für $k \in \mathbb{N}_0$ ist – sonst wäre a ja ein Quadrat. Damit ist

$$a^{(p-1)/2} = g^{(2k+1)(p-1)/2} = \underbrace{g^{k(p-1)}}_{=1} g^{(p-1)/2} \not\equiv_p 1,$$

weswegen wegen $g^{p-1} \equiv_p 1$ dann aber $a^{(p-1)/2} \equiv_p -1$ sein muss. □

Ein Zeuge für die Zerlegbarkeit ist nun eine Zahl a mit

$$a^{(p-1)/2} \not\equiv_p \left(\frac{a}{p}\right)$$

und der **Solovay–Strassen–Test** überprüft nun gerade zufällige Zahlen, ob sie bei diesem Primzahlkriterium scheitern.

4.3 Ein deterministischer Primzahltest

Die stochastischen Primzahltests sind ja schön und gut, haben aber natürlich den Nachteil, daß sie nie *Sicherheit* liefern, sondern die Ergebnisse nur mit großer bis sehr großer Wahrscheinlichkeit korrekt sind. Der große Nachteil *deterministischer* Primzahltests besteht in ihrer Komplexität. Man kann sich leicht überlegen, da sie beispielsweise beim Sieb des Erathostenes *exponentiell* mit der Größe der zu testenden Primzahl wächst und das macht nicht wirklich Spass.

Definition 4.21 *Die* ***Größe einer Zahl*** n *ist die Anzahl ihrer Stellen, also* $\log n$*.*

„Gute“ Algorithmen sind nun besser als exponentiell, idealerweise **polynomial** in der Größe der Zahl, haben also einen Rechenaufwand der Größenordnung $O\left((\log n)^k\right)$ für ein passendes k. Das wäre dann **polynomiale Komplexität**. Ein erster Primzahltest mit polynomialer Komplexität wurde 2004 von Agrawal, Kayal und Saxena veröffentlicht[112] (Agrawal *et al.*, 2004) und wird heute gerne als **AKS–Algorithmus** bezeichnet. Den wollen wir uns jetzt ansehen, die Darstellung selbst ist wieder einmal aus (Willems, 2008).

Dazu brauchen wir zuerst einmal eine sehr einfache und elementare Beobachtung.

[112] Der Titel lässt ahnen, daß bis dahin Testen auf Primalität als möglicher Kandidat für ein NP–Problem gesehen wurde, was uns zeigt, daß man sich auf Komplexität nicht verlassen sollte ...

Lemma 4.22 *Seien* $a, n \in \mathbb{N}$ *mit* $\mathrm{ggT}(a,n) = 1$ *und* $n \neq 1$. *Dann ist*

$$n \in \mathbb{P} \qquad \Leftrightarrow \qquad (x+a)^n \equiv_n x^n + a, \tag{4.18}$$

wobei die Aussage auf der rechten Seite als Aussage für Polynome zu lesen ist.

Beweis: Ist $n \in \mathbb{P}$, dann liefern binomische Entwicklung und kleiner Fermat

$$(x+a)^n = x^n + a^n + \sum_{j=1}^{n-1} \underbrace{\binom{n}{j}}_{\equiv_n 0} x^j a^{n-j} \equiv_n x^n + a \underbrace{a^{n-1}}_{\equiv_m 1} \equiv_n x^n + a.$$

Ist hingegen $n \notin \mathbb{P}$, dann gibt es eine Primzahl $p \in \mathbb{P}$ mit $n = pn'$ und wir erhalten

$$\binom{n}{p} = \frac{n!}{p!(n-p)!} = \frac{n(n-1)\cdots(n-p+1)!}{p!} \not\equiv_n 0,$$

denn sonst gäbe es k, so daß

$$n(n-1)\cdot(n-p+1) = knp(p-1)! \qquad \Rightarrow \qquad p \mid \prod_{j=1}^{p-1}(n-j),$$

was den offensichtlichen Widerspruch

$$0 \equiv_p \prod_{j=1}^{p-1}(n-j) = \prod_{j=1}^{p-1}(pn'-j) \equiv_p \underbrace{(-1)^{p-1}}_{\equiv_p 1} \underbrace{\prod_{j=1}^{p-1} j}_{\not\equiv_p 0}$$

liefert. Wegen $\mathrm{ggT}(a,n) = 1$ ist aber auch $a^n \not\equiv_n 0$ und damit enthält das Polynom $(x+a)^n$ einen Anteil mit x^p, die rechte Seite von (4.18) gilt also für zusammengesetzte Zahlen nicht. □

Lemma 4.22 allein ist leider zu billig für einen Test mit polynomialer Laufzeit, da die Berechnung *aller* Koeffizienten von $(x+a)^n$ wieder zu viel Arbeit ist. Man verwendet viel mehr einen Trick, den man auch aus der Computeralgebra kennt und bei der wirklich schnellen Multiplikation ganzer Zahlen nach Schönhage und Strassen (Schönhage & Strassen, 1971) benutzt, siehe (Gathen & Gerhard, 1999; Sauer, 2001). Und zwar betrachtet man jetzt den **Polynomring** $\mathbb{Z}_n[x]$ und interessiert sich nur noch für die Indentität

$$(x+a)^n \equiv_{x^r-1} x^n + a, \qquad \text{genauer} \qquad (x+a)^n \equiv_{x^r-1,\mathbb{Z}_n[x]} x^n + a, \tag{4.19}$$

wobei r noch passend zu wählen ist. Für diese Situation kann man nun die folgende Aussage treffen.

Satz 4.23 (AKS–Kriterium) *Sei* $n > 2$ *und* $r \in \mathbb{N}$ *mit* $\mathrm{ggT}(r,n) = 1$. *Ferner sei* $s \in \mathbb{N}$ *eine Zahl mit* $\mathrm{ggT}(a,n) = 1$, $a \in \mathbb{Z}_s + 1$ *und es gelte*[113]

$$\binom{\varphi(r)+s-1}{s} > n^{2d\left\lfloor\sqrt{\varphi(r)/d}\right\rfloor}, \qquad d \mid \frac{\varphi(r)}{t}, \tag{4.20}$$

[113] So etwas nennt man unter Fachleuten eine *technische Voraussetzung*.

wobei t *die Ordnung von* n *in* $\mathbb{Z}_r^\times$ *ist, d.h.* $n^t \equiv_r 1$.

Gilt nun

$$(x+a)^n \equiv_{x^r-1,\mathbb{Z}_n[x]} x^n + a, \qquad a \in \mathbb{Z}_s + 1, \tag{4.21}$$

dann ist n *eine Primzahlpotenz.*

Der Beweis wird ein wenig aufwendiger werden. Dazu nehmen wir jetzt immer an, daß a, n sowie r, s, t die Voraussetzungen von Satz 4.23 erfüllen. Damit das Verfahren später eine sinnvolle Verallgemeinerung liefert, ist außerdem $r < n$ sinnvoll.

Lemma 4.24 *Für jedes* a*, das* (4.21) *erfüllt und jeden Primteiler* p *von* n *gilt*

$$(x^m + a)^{n^j p^k} \equiv_{x^r-1,\mathbb{Z}_p[x]} x^{mn^j p^k} + a, \qquad m, j, k \in \mathbb{N}. \tag{4.22}$$

Beweis: Wir zeigen zuerst, daß für $j \in \mathbb{N}$ auch

$$(x+a)^{n^j} \equiv_{x^r-1,\mathbb{Z}_n[x]} x^{n^j} + a \tag{4.23}$$

gilt, und zwar durch Induktion über j, wobei der Fall $j = 1$ gerade (4.21) ist. Da für $\ell \in \mathbb{N}$

$$x^{r\ell} - 1 = (x^r - 1) \sum_{q=0}^{\ell-1} x^{qr} \qquad \Rightarrow \qquad x^r - 1 \,|\, x^{r\ell} - 1$$

gilt, erhalten wir für den Induktionsschritt

$$\begin{aligned}
(x+a)^{n^{j+1}} &= \left((x+a)^{n^j}\right)^n \equiv_{x^r-1,\mathbb{Z}_n[x]} (x^{n^j} + a)^n \\
&= x^{n^{j+1}} + a^n + \sum_{\ell=1}^{r-1} \binom{n}{\ell} x^{n^j \ell} a^{n-\ell} + x^{n^j r} \underbrace{\sum_{\ell=0}^{n-r-1} \binom{n}{\ell+r} x^{n^j \ell} a^{n-\ell-r}}_{=:f(x^{n^j})} \\
&= x^{n^{j+1}} + a^n + (x^{n^j r} - 1) f(x^{n^j}) + \sum_{\ell=1}^{r-1} \underbrace{\binom{n}{\ell}}_{\equiv_n 0} x^{n^j \ell} a^{n-\ell} + \sum_{\ell=r}^{n-1} \underbrace{\binom{n}{\ell}}_{\equiv_n 0} x^{n^j(\ell-r)} a^{n-\ell} \\
&= x^{n^{j+1}} + \underbrace{a^n}_{\equiv_n a} + \underbrace{(x^{n^j r} - 1)}_{\equiv_{x^r-1} 0} f(x^{n^j}) + n g(x^{n^j}) \equiv_{x^r-1,\mathbb{Z}_n[x]} x^{n^{j+1}} + a,
\end{aligned}$$

was uns sofort auch

$$(x+a)^{n^j} \equiv_{x^r-1,\mathbb{Z}_p[x]} x^{n^j} + a \tag{4.24}$$

liefert[114]. Nach Lemma 4.22 ist dann auch

$$(x+a)^{n^j p^k} \equiv_{x^r-1,\mathbb{Z}_p[x]} \left(x^{n^j} + a\right)^{p^k} \equiv_{x^r-1,\mathbb{Z}_p[x]} x^{n^j p^k} + a$$

[114] Das ist das rekurrente Standardargument, daß aus $a|b$ und $x \equiv_b y$ mittels $a|b|(x-y)$ auch $x \equiv_a y$ folgt.

und wenn wir in dieser Gleichung nun noch x durch x^m ersetzen, dann erhalten wir

$$(x+a)^{n^j p^k} \equiv_{x^{mr}-1,\mathbb{Z}_p[x]} x^{mn^j p^k} + a$$

und da $x^r - 1 | x^{rm} - 1$, folgt endlich auch (4.22). □

Übung 4.5 Zeigen Sie:

$$\binom{n}{k} \equiv_n 0, \qquad k = 1, \ldots, n-1.$$

♢

Definition 4.25 (Untergruppen) *Für $A \subseteq \mathbb{Z}_r^\times$ bezeichne*

$$U_A := \left\{ \prod_{a \in A} a^{k_a} \ : \ k_a \in \mathbb{N}_0 \right\}$$

die von A erzeugte Untergruppe von $\mathbb{Z}_r^\times$. Ist $A = \{a\}$, so schreiben wir einfach U_a.

Jetzt kommt wieder ein bisschen Gruppentheorie. Wir setzen $d = \frac{\#\mathbb{Z}_r^\times}{\#U_{\{p,n\}}}$, was ja nach dem Satz von Lagrange eine ganze Zahl sein muss. Nun ist $t := \#U_n$ ein Teiler von $\#U_{\{p,n\}}$, da U_n offensichtlich eine Untergruppe von $\#U_{\{p,n\}}$ ist und deswegen gilt

$$d = \frac{\#\mathbb{Z}_r^\times}{\#U_{\{p,n\}}} \Big| \frac{\#\mathbb{Z}_r^\times}{\#U_n} = \frac{\varphi(r)}{t}. \tag{4.25}$$

Schließlich seien noch $m_1, \ldots, m_d$ **Nebenklassenvertreter** von $U_{\{p,n\}}$ in $\mathbb{Z}_r^\times$, d.h.

$$\mathbb{Z}_r^\times = \bigcup_{j=1}^{d} m_j \, U_{\{p,n\}}. \tag{4.26}$$

Lemma 4.26 *Es gibt Zahlen $0 \le j, k, j', k' \le \left\lfloor \sqrt{\varphi(r)/d} \right\rfloor$ mit $(j,k) \neq (j',k')$, so daß*

$$n^j p^k =: u \equiv_r v := n^{j'} p^{k'}, \qquad (x^{m_k} + a)^u \equiv_{x^r-1,\mathbb{Z}_p[x]} (x^{m_k} + a)^v, \quad k = 1, \ldots, d,$$

mit den m_k aus (4.26) *erfüllt ist.*

Beweis: Es gibt

$$\left(\frac{\varphi(r)}{d} + 1 \right)^2 > \frac{\varphi(r)}{d} = t$$

mögliche Pärchen (j,k) und (j',k'), und da die Gruppe $U_{\{p,n\}}$ nur t Elemente hat, muss es mindestens zwei verschiedene (j,k) und (j',k') geben, so daß $(n^j p^k)_r = \left(n^{j'} p^{k'}\right)_r$ ist. Das ist die erste Hälfte der Behauptung.

Da es ein $\ell \in \mathbb{Z}$ gibt, so daß $x^u = x^{v+\ell r} \equiv_{x^r-1} x^v$ ist[115], erhalten wir aus Lemma 4.24, daß

$$(x^{m_k} + a)^u \equiv_{x^r-1,\mathbb{Z}_p[x]} x^{m_k u} + a \equiv_{x^r-1,\mathbb{Z}_p[x]} x^{m_k v} + a \equiv_{x^r-1,\mathbb{Z}_p[x]} (x^{m_k} + a)^v$$

[115] Wegen $x^{v+\ell r} - x^v = x^v(x^{\ell r} - 1) \equiv_{x^r-1} 0$.

wie behauptet. □

Jetzt betritt die Zahl s mit $\mathrm{ggT}(\mathbb{Z}_s+1, n) = \{1\}$ die Bühne. Da $p|n$, also $p \notin \mathbb{Z}_s+1$ ist natürlich $p > s$ und damit sind die linearen Polynome $x+a \in \mathbb{Z}_p[x]$, $a \in \mathbb{Z}_s+1$, alle verschieden. Nun definieren wir die Polynome

$$f_e(x) = \prod_{a=1}^{s}(x+a)^{e_a} \in \mathbb{Z}_p[x], \qquad e \in \mathbb{N}_0^s, \tag{4.27}$$

und stellen fest, daß $e \neq e'$ auch $f_e \neq f_{e'}$ impliziert, da die Linearfaktoren alle unterschiedlich sind[116]. Schließlich sei ζ eine primitive r–te **Einheitswurzel** in $\mathbb{Z}_p$, d.h. $\zeta^r \equiv_p 1 \not\equiv_p \zeta^k$, $k < r$, mit deren Hilfe wir einen Körper $\mathbb{K} \supseteq \mathbb{Z}_p$ der Ordnung r bauen können, siehe (Gathen & Gerhard, 1999). Solche Körper sind sehr nützlich in der Computeralgebra und liefern ein Konzept, das man eigentlich aus einem ganz anderen Kontext kennen könnte.

Definition 4.27 (DFT) *Die **diskrete Fouriertransformation** bzw. **DFT** von f_e ist der Vektor*

$$\widehat{f_e} := (f_e(\zeta^{m_k}) \;:\; k = 1, \ldots, d) \in \mathbb{K}^d.$$

Lemma 4.28 *Ist $e \neq e'$ und $\deg f_e, \deg f_{e'} < \varphi(r)$, so gilt $\widehat{f_e} \neq \widehat{f_{e'}}$, die Abbildung $e \mapsto \widehat{f_e}$ ist also surjektiv.*

Beweis: Nach Lemma 4.24 ist

$$\begin{aligned}(f_e(x^{m_\ell}))^{n^j p^k} &= \prod_{a=1}^{s}(x^{m_\ell}+a)^{e_a n^j p^k} \\ &\equiv_{x^r-1,\mathbb{Z}_p[x]} \textstyle\prod_{a=1}^{s}\left(x^{m_\ell n^j p^k}+a\right)^{e_a} \equiv_{x^r-1,\mathbb{Z}_p[x]} f_e(x^{m_\ell n^j p^k})\end{aligned}$$

Da $\zeta^r \equiv 1$ ist, also ζ eine Nullstelle von x^r-1 in $\mathbb{Z}_p$ ist, gilt diese Identität auch in $\mathbb{Z}_p$, das heißt, $(f_e(\zeta^{m_\ell}))^{n^j p^k} = f_e(\zeta^{m_\ell n^j p^k})$ in $\mathbb{Z}_p$.

Sind nun e, e' Vektoren mit $\widehat{f_e} = \widehat{f_{e'}}$, dann gilt für $\ell = 1, \ldots, d$ und j, k die Identität $f_e(\zeta^{m_\ell}) = f_{e'}(\zeta^{m_\ell})$ und damit

$$f_e(\zeta^{m_\ell n^j p^k}) = (f_e(\zeta^{m_\ell}))^{n^j p^k} = (f_{e'}(\zeta^{m_\ell}))^{n^j p^k} = f_{e'}(\zeta^{m_\ell n^j p^k}),$$

also hat $g := f_e - f_{e'}$ die Nullstellen $\zeta^{m_\ell n^j p^k}$. Wegen (4.26) kann man jedes $q \in \mathbb{Z}_r^\times$ durch geeignete Wahl von ℓ, j, k erhalten, so daß also g mindestens $\#\mathbb{Z}_r^\times = \varphi(r)$ Nullstellen hat. Der Grad von g ist aber höchstens $\varphi(r)-1$, so daß g nur das Nullpolynom sein kann. □

Beweis von Satz 4.23: Die Menge aller Multiindizes $e \in \mathbb{N}_0^s$ mit $\deg f_e = |e| := e_1+\cdots+e_s \leq \varphi(r)-1$ hat Kardinalität $\binom{s+\varphi(r)-1}{s}$, wie man sich mit einem einfachen Urnenmodell leicht überlegt. Nun sei G die von allen $(\zeta^{m_\ell}+a)$, $\ell = 1, \ldots, d$,

[116] Schlielich gibt e_a die Vielfachheit der Nullstelle von f_e an der Stelle $-a$ an und diese Werte bestimmen die Polynome eindeutig. Das ist **Hermite–Interpolation**, die Verallgemeinerung der Interpolation wie wir sie in Lemma 2.35 betrachtet haben.

$a = 1, \ldots, s$, erzeugte Untergruppe von $\mathbb{K}^\times$ und $G_0 = G \cup \{0\}$. Da $\widehat{f_e} \in G_0^d$ und da es mindestens $\binom{s+\varphi(r)-1}{s}$ verschiedene solche Vektoren gibt, ergibt sich

$$(\#G_0)^d = \#G_0^d \geq \binom{s+\varphi(r)-1}{s} > n^{2d\left\lfloor\sqrt{\varphi(r)/d}\right\rfloor}$$

nach der Voraussetzung (4.20). Daraus folgt aber sofort

$$\#G_0 > n^{2\left\lfloor\sqrt{\varphi(r)/d}\right\rfloor}. \tag{4.28}$$

Wählen wir nun wieder $0 \leq j, k, j', k' \leq \left\lfloor\sqrt{\varphi(r)/d}\right\rfloor$ wie in Lemma 4.26, dann ist

$$0 \leq u = n^j p^k < n^{j+k} \leq n^{2\left\lfloor\sqrt{\varphi(r)/d}\right\rfloor}, \qquad 0 \leq v \leq n^{2\left\lfloor\sqrt{\varphi(r)/d}\right\rfloor},$$

also

$$|u - v| \leq n^{2\left\lfloor\sqrt{\varphi(r)/d}\right\rfloor} \leq \#G_0 - 1 = \#G. \tag{4.29}$$

Nach Lemma 4.26 ist dann $(\zeta^{m_\ell} + a)^u \equiv_{x^r-1,\mathbb{Z}_p[x]} (\zeta^{m_\ell} + a)^v$ für alle ℓ und a und da ζ eine r–te Einheitswurzel ist, gilt dies auch wieder im strikten Sinne $(\zeta^{m_\ell} + a)^u = (\zeta^{m_\ell} + a)^v$. Anders gesagt:

$$g^u = g^v, \qquad g \in G, \tag{4.30}$$

und für $g = 0$ ist (4.30) trivialerweise erfüllt. Damit hat die Gleichung $x^u = x^v$ in $\mathbb{K}^* \supseteq G$ mindestens $\#G_0 = |u - v| + 1$ Lösungen, was nur für $u = v$ möglich ist – andernfalls hätte $g^{u-v} = 1$ zu viele Lösungen. Ist nun $j = j'$, dann bedeutet $n^j p^k = u = v = n^j p^{k'}$, daß auch $k' = k$ ist, was einen Widerspruch zur Aussage von Lemma 4.26 darstellt, denn dort war ja $(j, k) \neq (j', k')$. Ist hingegen $j \neq j'$, sagen wir $j > j'$, dann ist

$$n^{j-j'} = p^{k'-k} \qquad \Rightarrow \qquad n = p^{\frac{k'-k}{j-j'}},$$

also $j - j' \mid k' - k$ und n ist eine Primzahlpotenz. □

So, das war jetzt einmal ein etwas anstrengenderer Beweis, bleibt nur die Frage, wie wir das Ganze noch in einen Algorithmus packen.

Algorithmus 4.29 (AKS–Test)
Gegeben: $2 < n \in \mathbb{N}$.

1. *Ist* $n \in 2^{\mathbb{N}}$ *eine Zweierpotenz*[117]*, dann ist* n *zusammengesetzt.* ***Ende***
2. *Berechne*
$$N = 2n(n-1)(n^2-1)\cdots(n^{4\lceil\log_2 n\rceil^2} - 1) \tag{4.31}$$
und finde die kleinste Primzahl r*, die* N ***nicht*** *teilt.*
3. *Ist*[118] $n = p \in \mathbb{P}$ *für ein* $p < r$*, dann ist* n *eine Primzahl.* ***Ende***

[117] Das ist in einer Binärdarstellung unglaublich schwer zu überprüfen!
[118] Zufällig ...

4. *Gilt* $p \mid n$ *für ein* $p < r$, *dann ist* n *zusammengesetzt.* ***Ende***

5. *Gilt für ein* $a \in \{1, \ldots, r\}$

$$(x + a)^n \not\equiv_{x^r-1, \mathbb{Z}_n[x]} x^n + a,$$

dann ist n *zusammengesetzt.* ***Ende***

6. *Ist* $\sqrt[u]{n} \in \mathbb{Z}$ *für ein* $0 < u \leq \log_r n$, *dann ist* n *zusammengesetzt.* ***Ende***

Ergebnis: n *ist Primzahl.*

Bei der Analyse des Verfahrens müssen wir uns um zwei Dinge kümmern: Einmal, daß das Verfahren das tut, was es soll, also korrekt ist, und dann natürlich auch, daß das Verfahren wirklich nur einen polynomialen Aufwand in $\nu := \log n$ hat. Dazu sehen wir uns die einzelnen Schritte von Algorithmus 4.29 genauer an.

Der Test aus 1) ob n eine Zweierpotenz ist, ist sehr einfach durchzuführen, man muss schließlich nur testen, ob n in Binärdarstellung von der Form $10\ldots0$ ist, was man mit $\log_2 n$ Suchoperationen durchführen kann, indem man sich die Bits von hinten nach vorne ansieht. Daß man im Falle eine Zweierpotenz fertig ist, versteht sich natürlich von selbst, ebenso, wenn auch nur einige binäre Endziffern von n den Wert 0 haben. Danach setzt man ν formal als

$$\nu := \left\lceil \log_2 n \right\rceil, \tag{4.32}$$

so daß $2^{\nu-1} < n < 2^\nu$ garantiert ist.

Schritt 2 ist da schon ein wenig aufwendiger. Immerhin hat das Produkt in (4.31) $4\nu^2 + 2$ Terme[119] und der Aufwand der Multiplikation ist ein Polynom in der Größe, siehe das **Karatsuba**–Verfahren in (2.3). Der größte Faktor ist

$$n^{4\nu^2} - 1 \simeq n^{4\nu^2} = 2^{4 \log_2 n\, \nu^2} \simeq 2^{4\nu^3},$$

und dessen Länge ist damit $O(\nu^3)$, also polynomial in ν. Die Länge von N ist

$$\begin{aligned} \log_2 N &\leq 1 + \log_2 n + \sum_{j=1}^{4\nu^2} \log_2(n^j - 1) \leq 1 + \log_2 n + \sum_{j=1}^{4\nu^2} j \log_2 n \\ &\leq 1 + \nu + \nu \sum_{j=1}^{4\nu^2} j = 1 + \nu\left(1 + \frac{4\nu^2(4\nu^2+1)}{2}\right), \end{aligned}$$

und damit ist die Länge $\kappa := \left\lceil \log_2 N \right\rceil$ ebenfalls polynomial in ν beschränkt. Damit ist N mit polynomialem Aufwand berechenbar. Nach einem Satz von Tscheyscheff[120], so zitiert in (Willems, 2008), ist

$$\prod_{p \in \mathbb{P}, p < 2k} p > 2^k, \qquad k \geq 2,$$

[119] Wenn man von der Multiplikation mit 2 einmal absieht.

[120] Der Mann mit den vielen Transkriptionen, auch Chebychev, Chebychov, Tschebyschev, Tschebyschov, ...

so daß das Produkt der ersten 2κ Primzahlen einen Wert größer als N ergibt und mindestens eine dieser Primzahlen kann damit N nicht teilen. Die kleinste unter diesen Primzahlen wird unser r. Wir müssen also maximal 2κ Divisionen mit Rest durchführen, um r zu finden und da nach Bemerkung 2.11 dies im wesentlichen mit demselben Aufwand wie die Multiplikation durchführen lässt, bleiben wir immer noch polynomial. Das geht sogar mit Erathostenes.

Bemerkung 4.30 *In der Praxis tabelliert man die Primzahlen* $\mathbb{P}_{\leq k} := \{p \in \mathbb{P} \,:\, p \leq k\}$ *bis zu einer gewissen Ordnung im Voraus und zwar in der Zahl*

$$P_{\leq k} := \prod_{p \in \mathbb{P}_{\leq k}} p.$$

Der Test, ob eine Primzahl aus $\mathbb{P}_{\leq k}$ *Teiler von* $n \in \mathbb{N}$ *ist, besteht dann aus der relativ einfachen Berechnung von* $\mathrm{ggT}(P_{\leq k}, n)$*, der gleich das Produkt* aller *Primteiler aus* $\mathbb{P}_{\leq k}$ *von* n *findet.*

Die Schritte 3) und 4) in Algorithmus 4.29 erledigt man dadurch mit, daß man bei jedem Test von N auf Teilbarkeit durch ein $p \in \mathbb{P}_{\leq 2\kappa}$ auch gleich n auf Teilbarkeit durch p überprüft. Da $n \ll N$ ist, kostet das praktisch nichts extra.

Der interessante Teil ist natürlich Schritt 5), alles andere ist eigentlich nur Vorgeplänkel. Dabei bemerken wir zunächst, daß die Ordnung t von n in $\mathbb{Z}_r^\times$ größer als $4\nu^2$ ist. Wäre das nämlich nicht der Fall, dann gäbe es ein $k \leq 4\nu^2$ mit $n^k \equiv_r 1$, also $r \mid n^k - 1$ und da die $n^k - 1$ gerade die Faktoren von N sind, dann hieße das $r \mid N$, was wir aber in Schritt 2) gerade ausgeschlossen haben[121].

Lemma 4.31 *Setzt man mit den bisher berechneten Größen* $s = r$*, dann erfüllen* r, s, t *die Voraussetzungen von Satz 4.23.*

Beweis: Die Teilbarkeitsforderungen $\mathrm{ggT}(r, n) = \mathrm{ggT}(\mathbb{Z}_s + 1, n) = 1$ sind per Konstruktion erfüllt. Bleibt also (4.20). Da $t > 4\nu^2$ ist, haben wir

$$d \leq \frac{\varphi(r)}{t} < \frac{\varphi(r)}{4\nu^2}$$

und somit

$$2d\left\lfloor \sqrt{\frac{\varphi(r)}{d}} \right\rfloor \leq 2d\sqrt{\frac{\varphi(r)}{d}} = \sqrt{4d\varphi(r)} < \sqrt{\frac{\varphi(r)^2}{\nu^2}} = \frac{\varphi(r)}{\nu} < \frac{\varphi(r)}{\log_2 n}. \tag{4.33}$$

Da N gerade ist, ist $r \geq 3$ und da $\varphi(r) = r - 1 \geq 2$ ist, ergbit sich also

$$\binom{\varphi(r) + s - 1}{s} = \binom{\varphi(r) + r - 1}{r} = \binom{2\varphi(r)}{\varphi(r) + 1} = \frac{(2\varphi(r)) \cdots (\varphi(r) + 2)}{(\varphi(r) - 1) \cdots 1} \geq 2^{\varphi(r)},$$

und unter Einsetzen von (4.33), von rechts nach links gelesen,

$$\binom{\varphi(r) + s - 1}{s} \geq \left(n^{\log_2 n}\right)^{\varphi(r)} > n^{2d\left\lfloor \sqrt{\frac{\varphi(r)}{d}} \right\rfloor},$$

[121] Das erklärt dann auch die zuerst etwas seltsam erscheinende Konstruktion von N.

wie in (4.20) gefordert. □

Wir können also endlich das AKS–Kriterium aus Satz 4.23 anwenden. Das sagt uns, daß für jede Nicht–Primzahlpotenz der Test (4.21) irgendwann scheitern muss und genau danach sucht Schritt 5). Polynomialität wird hierbei durch die *Gradschranke* $r \leq 2\kappa$ gewährleistet und das ist der Punkt, der letztlich den ganzen Extraufwand rechtfertigt.

In Schritt 6) wird schliesslich noch – ebenfalls polynomial – der Fall erledigt, daß n keine Primzahl sondern „nur" eine Primzahlpotenz ist.

Bemerkung 4.32 *Das war jetzt mal richtig Mathematik und auch durchaus aufwendig. Aber das zeigt auch, wie schwer es ist, Algorithmen anzugeben, die Primalität in polynomialer Zeit testen. Wenn man alles zusammenzählt, kommt man auf* $O(\nu^{6+\varepsilon})$*, was schon recht gut aber fr praktische Zwecke glücklicherweise immer noch ziemlich schlecht ist: Eine Verdopplung der Primzahllänge sorgt immer noch für 64–fachen Aufwand beim Testen, eine Skalierung, die sich nicht so einfach erreichen lässt.*

Und da jede Primfaktorzerlegung immer auch ein Primzahltest ist, wenn auch ein etwas überdimensionierter, besteht immer noch Hoffnung auf weitgehende Sicherheit von Primzahlprodukten.

Bemerkung 4.33 *In (Gathen & Gerhard, 1999, 18.6) wird als Weltrekord für ein deterministisches Verfahren zur Bestimmung der Primalität noch ein Verfahren von Adleman, Pommerance und Rumely (Adleman* et al.*, 1983) angegeben, das auf eine Komplexität von* $\nu^{O(\log\log\nu)}$ *kommt, was einer polynomialen Laufzeit in der Praxis schon recht nahe kommt, denn die* „log log"*–Funktion ist schon ziemlich konstant.*

4.4 Finden von Primzahlen

Verschlüsselungsverfahren wie RSA leben davon, daß man eine große Primzahl verwendet, die aber natürlich erst einmal gefunden werden will. Konkret:

> *Für eine vorgegebene Länge* $\nu \in \mathbb{N}$ *finde eine Primzahl* p *dieser Länge, also* $\log_2 p = \nu$.

Die Verfahren sind wieder randomisierter Natur und verwenden Wahrscheinlichkeiten auf Basis des Primzahlsatzes 4.3, allerdings in einer quantitativen Form wie dem Satz von Tschebyscheff, Satz 4.4. In (Gathen & Gerhard, 1999, Theorem 18.7) findet sich dazu die folgende Abschätzung:

$$\frac{x}{\log x}\left(1+\frac{1}{2\log x}\right) < \pi(x) < \frac{x}{\log x}\left(1+\frac{3}{2\log x}\right), \qquad x \geq 59, \tag{4.34}$$

wieder unter Verwendung des natürlichen Logarithmus. Das ist gut genug für uns, denn die paar Primzahlen kleiner als 59 fallen nicht wirklich ins Gewicht.

Suchen wir nun eine Primzahl p einer bestimmten binären Länge, dann bedeutet dies[122], für vorgegebenes $B \in \mathbb{N}$ nach $B < p \leq 2B$ zu suchen, mit $\lfloor \log_2 B \rfloor = \nu$.

[122] Ein Bit hin oder her

Algorithmus 4.34 (Primzahlsuche)
Gegeben: $B, k \in \mathbb{N}$.

1. *Wähle **zufällig** eine Zahl* $n \in [B+1, 2B]$.
2. *Führe maximal* k *Versuche mit einem probabilistischen Primzahltest, beispielsweise dem Fermat–Test 4.11, durch.*
3. *Ergibt sich „weiß nicht", dann akzeptiere* n *als Primzahl, andernfalls wiederhole 1).*

Ergebnis: *Eine Zahl* n*, die wahrscheinlich eine Primzahl ist.*

Tatsächlich ist n sogar sehr wahrscheinlich eine Primzahl.

Proposition 4.35 *Die Wahrscheinlichkeit, daß* n *in Algorithmus 4.34 eine Prinzahl ist, ist* $\geq 1 - 2^{-k+1} \log B$.

Beweis: Für $56 \leq x =: e^{\xi}$, also $\xi \geq 5$, erhalten wir aus (4.34), daß

$$\begin{aligned}
&\pi(2x) - \pi(x) \\
&> \frac{2x}{\log 2x}\left(1 + \frac{1}{2\,\log 2x}\right) - \frac{x}{\log x}\left(1 + \frac{3}{2\,\log x}\right) \\
&= \frac{x}{\xi}\left(\frac{2\xi}{\xi + \log 2} + \frac{2\xi}{2(\xi+\log 2)^2} - 1 - \frac{3}{2\,\xi}\right) > \frac{x}{\xi}\left(\frac{2\xi}{\xi+1} + \frac{2\xi}{2(\xi+1)^2} - 1 - \frac{3}{2\,\xi}\right) \\
&= \frac{x}{\xi}\left(\frac{2\xi(\xi+1)+\xi}{(\xi+1)^2} - 1 - \frac{3}{2\,\xi}\right) = \frac{x}{\xi}\left(\frac{2(\xi+1)^2 - (\xi+1)}{(\xi+1)^2} - 1 - \frac{3}{2\,\xi}\right) \\
&= \frac{x}{\xi}\left(1 - \frac{3(\xi+1)+2\xi}{2\xi(\xi+1)}\right) = \frac{x}{\xi}\left(1 - \frac{5(\xi+1)-2}{2\xi(\xi+1)}\right) > \frac{x}{\xi}\left(1 - \frac{5}{2\xi}\right) \\
&= \frac{x}{\xi}\,\frac{2\xi-5}{2\xi} > \frac{x}{\xi}\,\frac{\xi}{2\xi} = \frac{x}{2\xi},
\end{aligned}$$

also[123]

$$\pi(2x) - \pi(x) > \frac{x}{2\,\log x}. \tag{4.35}$$

Aus diesem Grund ist

$$\#\left(\mathbb{P} \cap [B+1, 2B]\right) = \pi(2B) - \pi(B) > \frac{B}{2\log B}, \tag{4.36}$$

und damit für $n \in [B+1, 2B]$

$$P(n \in \mathbb{P}) = \frac{\#\left(\mathbb{P} \cap [B+1, 2B]\right)}{B} > \frac{1}{2\log B}. \tag{4.37}$$

[123] Siehe (Gathen & Gerhard, 1999, Exercise 18.18).

Sei C das Ereignis $n \notin \mathbb{P}$ und T das Ereignis „weiß nicht“, dann ist $P(n \in \mathbb{P}) \leq P(T)$ und daher

$$\begin{aligned}\frac{P(C|T)}{2\log B} &< P(n \in \mathbb{P})\, P(C|T) \leq P(T)\, P(C|T) = P(C \cap T) = P(T|C)\, P(C) \\ &\leq P(T|C) \leq 2^{-k},\end{aligned}$$

die Wahrscheinlichkeit für ein fehlerhaftes Ergebnis, also $P(C|T)$, ist somit kleiner als $2^{-k+1} \log B$, was genau die Behauptung liefert. □

Bemerkung 4.36

1. *Der Beweis von Proposition 4.35 und die große Wahrscheinlichkeit, daß man so eine Primzahl findet, funktioniert nur, weil es, wie* (4.37) *zeigt, doch recht wahrscheinlich ist, zwischen* $B + 1$ *und* $2B$ *über eine Primzahl zu stolpern.*

2. *Proposition 4.35 stellt auch eine Beziehung zwischen* B *und* k *her. Möchte man mit einer „Sicherheit“ von* $1 - \varepsilon$ *eine Primzahl bekommen, dann bedeutet das, daß*

$$2^{-k+1} \log B \leq \varepsilon \qquad \Leftrightarrow \qquad -k + 1 \leq \log_2 \varepsilon - \log_2 \log B,$$

   was zu

$$k \geq 1 + \log_2 \log B - \log_2 \varepsilon \tag{4.38}$$

   führt, wobei man beachten sollte, daß für vernünftige Werte von ε*, insbesondere* $\varepsilon \ll 1$*, auch* $\log_2 \varepsilon < 0$ *ist. Trotzdem ist* (4.38) *keine allzu massive Forderung an die Anzahl* k *der Iterationen im Primalitätstest, beispielsweise dem* ***Fermat–Test****.*

3. *Man kann auch den Erwartungswert für die benötigten Rechenoperationen*[124] *bestimmen, der laut (Gathen & Gerhard, 1999, Theorem 18.8) von der Größenordnung* $O\left(k\,(\log B)^2 M(\log B)\right)$ *ist, wobei* $M(n)$ *die Komplexität einer Multiplikation der Länge* n *bezeichnet, die nach Schönhage–Strassen (Schönhage & Strassen, 1971)* $O(n \log n)$ *beträgt. Insgesamt käme man dann also auf*

$$O\left(k\,(\log B)^3 \log\log B\right),$$

   und wer will, kann jetzt auch noch (4.38) *einsetzen.*

Das Zwischenergebnis (4.36) beweist das nachfolgende klassische Resultat für $n \geq 56$, für die kleineren Zahlen überprüft man es einfach von Hand oder per Computer.

Korollar 4.37 (Betrandsches Postulat) *Zwischen* n *und* $2n$*,* $n \in \mathbb{N}$*, liegt immer eine Primzahl.*

Übung 4.6 Überprüfen Sie das Bertrandsche Postulat für $n \leq 56$. ◊

[124] Im Sinne von „Wortoperationen“

4.5 Faktorisierung I: Pollard

Nachdem wir jetzt also einige Methoden kennen, um festzustellen, *ob* eine Zahl faktorisierbar ist, wird es langsam Zeit, diese Faktorisierung auch zu bestimmen.

Dabei gehen wir immer davon aus, daß wir bereits getestet haben, daß unsere Zahl n faktorisierbar ist[125] und daß n ungerade ist, denn das Abdividieren von Zweierpotenzen ist keine wirklich schwere Aufgabe.

Bemerkung 4.38 (Kleine Primfaktoren) *Zur Ermittlung* kleiner *Primfaktoren von n berechnet man die ersten k Primzahlen $p_1, \ldots, p_k$ vor*[126] *und dividiert diese ab, indem man das Produkt $m = p_1 \cdots p_k$ speichert und $\mathrm{ggT}(m, n)$ bildet. Dieser Wert enthält alle Faktoren in $p_1, \ldots, p_k$ und die ggT–Bestimmung kann wiederholt werden, um auch deren Potenzen abzudividieren. Diesen Berechnungsschritt kann man auch vor einem probabilistischen oder deterministischen Primzahltest durchführen, um erst einmal „Kleinkram" auszuschließen.*

Die naivste Methode ist natürlich die Probedivision aller (Prim-)Zahlen kleiner als $\sqrt{n}$, möglicherweise gekoppelt mit einem Primzahlsieb, aber das ist, wie man sich leicht vorstellen kann, zu langsam. Eine schnellere Faktorisierung liefert **Pollards $(p-1)$–Methode**, die folgendermaßen funktioniert.

Algorithmus 4.39 ($(p-1)$–Methode)
Gegeben: $n \in 2\mathbb{N}+1$ *zerlegbar.*

1. *Wähle* $2 \le a, b < n$.
2. *Ist* $\mathrm{ggT}(a, n) \neq 1$, *dann ist ein Teiler gefunden.*
3. *Berechne*
$$a_b = \left(a^{b!}\right)_n, \qquad a^{b!} = \left(\left(a^2\right)^3 \cdots\right)^b \tag{4.39}$$
und
$$d = \mathrm{ggT}(a_b - 1, n),$$
wobei alle Rechnungen in (4.39) *modulo n durchzuführen sind.*
4. *Ist $d \neq 1$, dann ist ein Teiler gefunden, ansonsten wähle neue a, b.*

Warum funktioniert nun dieses Verfahren? Sei $p \in \mathbb{P}$ ein Primteiler von n und

$$p - 1 = \prod_{j=1}^{k} p_j^{e_j}, \qquad p_j \in \mathbb{P}.$$

Ist nun $p_j^{e_j} \le b$ für alle $j = 1, \ldots, k$, dann folgt sofort, daß $p_j^{e_j} | b!$ und damit auch $(p-1)|b!$. Also ist nach dem kleinen Fermat

$$a^{b!} = a^{q(p-1)} = (a^q)^{p-1} \equiv_p 1, \qquad \text{d.h.} \qquad a_b - 1 \equiv_p 0$$

[125] Damit haben wir uns ja nun auch wirklich lange und intensiv genug beschäftigt.
[126] Beispielsweise mit dem Sieb des Erathostenes.

und damit $p | \mathrm{ggT}(a_b - 1, n)$. Das Fazit des Ganzen: Wir müssen eigentlich nur b groß genug machen, was aber auch den Aufwand massiv steigert.

Algorithmus 4.39 ist besonders erfolgreich und effizient, wenn $p - 1$ viele verschiedene Primteiler hat, denn dann kann man b relativ klein wählen – die Bedingung ist ja $p_j^{e_j} \leq b$. Produkte von Fermat–Zahlen sind andererseits sehr robust, da ja dann $p - 1$ selbst eine Primzahlpotenz ist und $b \geq p - 1$ gewählt werden muss.

Übung 4.7 Implementieren Sie den $(p - 1)$–Test und versuchen Sie, $f_3 \cdot f_4$ zu faktorisieren. ◊

Ein sehr einfaches und heuristisches, aber erstaunlich leistungsfähiges Verfahren ist **Pollards ρ–Methode**, (Pollard, 1975), die auf einer sehr einfachen Iteration basiert, und zwar auf der Funktion

$$f(x) = \left(x^2 + 1\right)_n , \tag{4.40}$$

wobei n die zu faktorisierende Zahl ist. Beginnt man das mit einem $x_0 \in \mathbb{Z}_n$, dann erhält man eine Folge $x_j = f^j(x_0)$, $j \in \mathbb{N}_0$. Ist nun p ein Primteiler von n, dann gibt es Werte $k, \ell \in \mathbb{N}$, so daß

$$x_{k+\ell} \equiv_p x_k,$$

wobei $k + \ell \leq p$ sein muss, denn nach spätestens p Schritten muss die erste Wiederholung auftreten. Andererseits, und dies ist das „Monte Carlo" an diesem Ansatz, sind die $(x_j)_{p_k}$ modulo der verschiedenen Primteiler p_k von n **unabhängige Zufallsvariablen** nach dem Chinesischen Restsatz und daher ist damit zu rechnen, daß für das obige Pärchen k, ℓ und für einen Primteiler p' von n eben auch $x_{k+\ell} \not\equiv_{p'} x_k$ erfüllt, also $x_{k+\ell} - x_k \not\equiv_n 0$ und dann liefert

$$p \,|\, \mathrm{ggT}(x_{k+\ell} - x_k, n)$$

einen nichttrivialen Teiler von n. Bleibt also nur noch die Bestimmung von k und ℓ, wobei die triviale Abschätzung $k + \ell \leq p < \sqrt{n}$ nahelegen würde, die Werte $x_0, \ldots, x_{\lfloor \sqrt{n} \rfloor}$ aufzuzeichnen und dann den ggT aller möglichen Pärchen zu bilden, was schon ein klein wenig ineffizient ist. Der Erwartungswert für $k + \ell$ liegt allerdings nach dem Geburtstagsparadoxon glücklicherweise bei $\sqrt{p}$.

Unabhängig davon gibt es einen netten und billigen Trick[127]: Man berechnet die beiden Folgen

$$x_{j+1} = f(x_j), \qquad y_{j+1} = f\left(f(y_j)\right) = x_{2j}, \qquad y_0 = x_0,$$

und untersucht $\mathrm{ggT}(x_j - y_j, n)$. Da[128]

$$x_{j+1} \equiv_n x_j^2 + 1 \qquad \Rightarrow \qquad x_{j+1} \equiv_p x_j^2 + 1,$$

erhalten wir, daß

$$x_{k+\ell} \equiv_p x_k \qquad \Leftrightarrow \qquad x_{k+r\ell} \equiv_p x_k, \quad r \in \mathbb{N}_0.$$

[127] In (Gathen & Gerhard, 1999, Algorithm 19.6) als *Floyd's cycle detection trick* bezeichnet.
[128] Standardargument ...

Das gilt dann aber nicht nur für das anfängliche k, bei dem zum ersten Mal $x_{k+\ell} \equiv_p x_k$ aufgetreten ist, sondern für *jedes* $k' > k$. Also bedeutet $y_j \equiv_p x_j$, daß $2j = j + r\ell$, also $j = r\ell$ und das passiert zum ersten Mal für

$$j = \begin{cases} \ell, & k = 0, \\ k + (-k)_\ell, & k \neq 0, \end{cases}$$

so daß mit maximal $k + \ell \leq p < n$ Schritten eine Kongruenz modulo p gefunden wird.

Algorithmus 4.40 (Pollards ρ–Methode)
Gegeben: $n \in \mathbb{N}$, *keine Primzahlpotenz.*

1. *Wähle* $x_0 \in \mathbb{Z}_n$.

2. *Für* $j = 1, \ldots, n$

   (a) *Berechne (modulo* n*)*

   $$x_{j+1} = f(x_j), \qquad y_{j+1} = f\left(f(x_j)\right), \qquad d_j = \mathrm{ggT}\left(x_j - y_j, n\right)$$

   (b) *Ist* $d_j \neq 1$*, dann ist* d_j *ein nichttrivialer Teiler von* n*.* ***Ende***

3. *Ergebnis: „Fehler“.*

Bemerkung 4.41 (ρ–Methode)

1. *Die große Stärke der ρ–Methode ist ihr Einfachheit. Alles was man braucht ist die einfache Funktion* f *und die Berechnung des* ggT*:*

2. *Der Fall „Fehler“ kann in Algorithmus 4.40 wirklich eintreten, nämlich dann, wenn in allen Fällen zufällig auch* $x_j \equiv_{p'} y_j$ *für alle anderen Primteiler* p' *von* n *ist. Das ist anscheinend nicht auszuschließen und laut (Gathen & Gerhard, 1999) gibt es auch keinen Beweis, daß die ρ–Methode wirklich immer funktioniert. Vieles ist da eher Heuristik.*

3. *Die Funktion* $f(x) = x^2 + 1$ *ist „. . . black magic . . . “ (Gathen & Gerhard, 1999) und funktioniert einfach gut. Lineare Funktionen erfüllen den Zweck nicht, die Werte gut zu „mischen“ und es sind keine Funktionen bekannt, die nachweisbar besser sind, also belässt man es bei dem einfachen quadratischen Polynom.*

Beispiel 4.42 (Faktorisierung) *Als kleines Beispiel faktorisieren wir einmal die Zahl* 4619863 *mit einer* `Octave`*–Implementierung des Verfahrens. Die Pollardmethode liefert hierfür*

j	x_j	y_j	d_j
1	*3080825*	*1866304*	*1*
2	*1866304*	*1256978*	*1*
3	*349923*	*4051317*	*1*
4	*1256978*	*3437569*	*1*
5	*546485*	*3559027*	*1*
6	*4051317*	*2690213*	*1*
7	*1979733*	*3018732*	*1*
8	*3437569*	*730985*	*1*
9	*1014116*	*1030486*	*1*
10	*3559027*	*3201629*	***83***

und 83 ist der gefundene Primteiler. Den übriggebliebenen Faktor 55661 liefert die Methode dann auch zurück, was uns die Primfaktorzerlegung 83×55661 vermuten lässt. Eigentlich müsste man über die beiden Faktoren aber noch einen Primzahltest laufen lassen, um sicherzugehen, aber `maxima` *bestätigt diese Faktorisierung, also scheint es zu stimmen.*

4.6 Faktorisierung II: Quadratische Kongruenzen

Daß quadratische Funktionen hilfreich sein können, haben wir ja bei der ρ–Methode oder im Kontext des Legendre–Symbols, Definition 4.19 gemerkt. Man kann quadratische Kongruenzen aber auch sehr schön für Faktorisierung verwenden. Die wesentliche Idee ist die folgende Aussage, bei der die Formulierung fast länger ist als der Beweis.

Lemma 4.43 *Wenn es Zahlen x, y gibt, die*

$$x \not\equiv_n \pm y, \qquad x^2 \equiv_n y^2, \tag{4.41}$$

dann ist $\mathrm{ggT}(x \pm y, n) \neq 1$.

Beweis: Aus der zweiten Identität in (4.41) folgt

$$n \,|\, x^2 - y^2 = (x+y)(x-y)$$

und da $x \pm y \not\equiv_n 0$, muß n zusammengesetzt und $\mathrm{ggT}(x \pm y, n) \neq 1$ sein. □

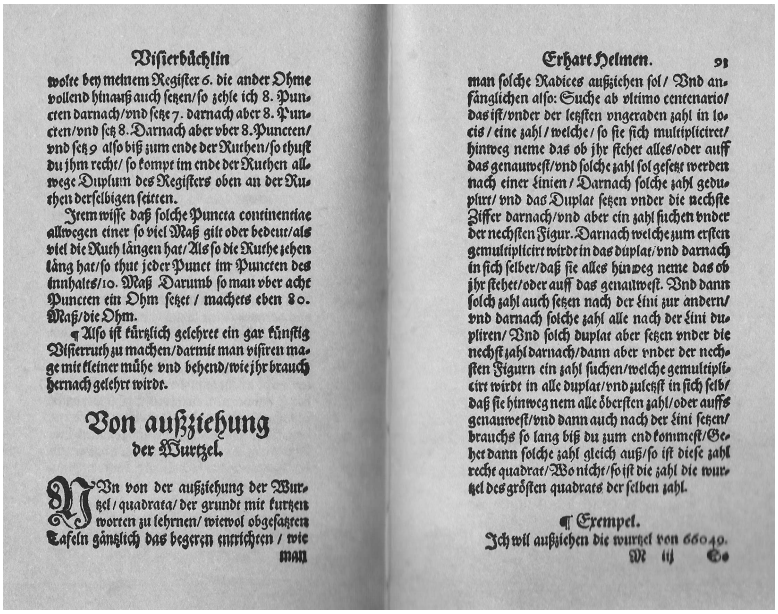
Die Idee, quadratische Kongruenzen oder Identitäten und Faktorisierungen als äquivalent anzusehen, d.h.

$$n = x^2 - y^2 = (x+y)(x-y) \qquad \text{und} \qquad n = xy = \left(\frac{x+y}{2}\right)^2 - \left(\frac{x-y}{2}\right)^2$$

ist klassisch und war bereits Fermat bekannt, der sie zum Faktorisieren nutzte: Zu n testet man für die Zahlen $m = \lfloor \sqrt{n} \rfloor, \lfloor \sqrt{n} \rfloor + 1, \ldots$ ob $m^2 - n$ ein Quadrat ist und nutzt dieses zur Faktorisierung.

Beispiel 4.44 *Fermat nutzte diesen Trick, um $n = 2027651281$ zu faktorisieren: Beginnt man mit $\lfloor \sqrt{n} \rfloor = 45029$ und probiert ein wenig, so ergibt sich*

$$45041^2 - n = 1020^2 \qquad \Rightarrow \qquad n = 45041^2 - 1020^2 = 46061 \cdot 44021.$$



Visierbüchlin

wolte bey meinem Register 6. die ander Ohme vollend hinauß auch setzen/so zehle ich 8. Puncten darnach/vnd setze 7. darnach aber 8. Puncten/vnd setz 8. Darnach aber vber 8. Puncten/ vnd setz 9 also biß zum ende der Ruthen/so thustu jhm recht/ so kompt im ende der Ruthen allwege Duplum des Registers oben an der Ruthen derselbigen seitten.

Item wisse daß solche Puncta continentiae allwegen einer so viel Maß gilt oder bedeut/als viel die Ruth längen hat/Als so die Ruthe zehen läng hat/so thut jeder Punct im Puncten des innhalts/10. Maß Darumb so man vber achte Puncten ein Ohm setzet / machets eben 80. Maß/die Ohm.

¶ Also ist kürtzlich gelehret ein gar künstig Visierruth zu machen/darmit man visiren mage mit kleiner mühe vnd behend/wie jhr brauch hernach gelehret wirde.

Von außziehung der Wurtzel.

NVn von der außziehung der Wurtzel / quadrata/ der grunde mit kurtzen worten zu lehrnen/ wiewol obgesatzten Tafeln gäntzlich das begeren entrichten / wie man

Erhart Helmen. 91

man solche Radices außziehen sol / Vnd anfänglichen also: Suche ab vltimo centenario/ das ist/vnder der letzsten vngeraden zahl in locis / eine zahl / welche / so sie sich multipliciret/ hinweg neme das ob jhr stehet alles/oder auff das genauwest/vnd solche zahl sol gesetzt werden nach einer Linien / Darnach solche zahl gedupliert/ vnd das Duplat setzen vnder die nechste Ziffer darnach/vnd aber ein zahl suchen vnder der nechsten Figur. Darnach welche zum ersten gemultiplicirt wirde in das duplat/vnd darnach in sich selber/daß sie alles hinweg neme das ob jhr stehet/oder auff das genauwest. Vnd dann solch zahl auch setzen nach der Lini zur andern/ vnd darnach solche zahl alle nach der Lini dupliren/ Vnd solch duplat aber setzen vnder die nechst zahl darnach/dann aber vnder der nechsten Figurn ein zahl suchen/welche gemultipliciirt wirde in alle duplat/vnd zuletzst in sich selb/ daß sie hinweg nem alle öbersten zahl/oder auffs genauwest/vnd dann auch nach der Lini setzen/ brauchs so lang biß du zum end kommest/Gehet dann solche zahl gleich auß/so ist diese zahl reche quadrat/Wo nicht/so ist die zahl die wurtzel des grösten quadrats der selben zahl.

¶ Exempel.

Ich will außziehen die wurtzel von 66049.

M iij So

Abbildung 4.1: Auszug aus dem Visierbüchlein des Erhart Helmen, das Bestandteil der späten Adam–Riese–Ausgabe (Risen & Helmen, 1574) ist (das „Original" ist etwa 1512 erschienen), mit der Anleitung zum Wurzelziehen. Viel Spass beim Ausprobieren.

Bemerkung 4.45 (Wurzelziehen) *Bleibt noch die Frage, wie man feststellt, ob eine Zahl Quadratzahl ist, aber das ist einfach: Man zieht die Wurzel. Verfahren dafür waren seit der Renaissance bereits reichlich bekannt, siehe Abb. 4.1 und als numerisches Verfahren steht ausserdem das* ***Heron–Verfahren*** *zur Verfügung, das sich bereits auf babylonischen Keilschrifttafeln findet und das auch in ganzzahliger Variante wunderbar funktioniert, siehe (Sauer, 2001). Man kann nämlich recht einfach zeigen, daß die Iterationsvorschrift*

$$x_0 = n, \qquad x_{j+1} = \left\lfloor \frac{x_j}{2} + \frac{n}{2x_j} \right\rfloor$$

nach endlich vielen *Schritten entweder* $x_j = \sqrt{n}$ *oder* $x_j^2 < n$ *erfüllt. Damit lässt sich systematisch ermitteln, ob eine gegebene Zahl eine Quadratzahl ist oder nicht.*

Natürlich ist die **Dixon–Methode** nicht so einfach wie die nette kleine Beobachtung in Lemma 4.43. Die grundlegende Idee besteht darin, **quadratische Kongruenzen** zu konstruieren und mit deren Hilfe Faktoren zu bestimmen.

Algorithmus 4.46 (Dixon) ***Gegeben:*** $n \in \mathbb{N}$.

1. *Bestimme eine* ***Faktorbasis***

$$F = \{p_1, \ldots, p_r\} \subset \mathbb{P}$$

von r *Primzahlen.*

2. *Für*[129] $s > r$ *finde Zahlen* $x_1, \ldots, x_s$ *mit*

$$x_j^2 \equiv_n p_1^{e_{j1}} \cdots p_r^{e_{jr}}, \qquad j = 1, \ldots, s. \tag{4.42}$$

3. *Für* $j = 1, \ldots, s$ *setze*

$$a_j = (e_{j1}, \ldots, e_{jr})_2 := ((e_{j1})_2, \ldots, (e_{jr})_2)$$

und finde $S \subset \{1, \ldots, s\}$ *mit*

$$\sum_{j \in S} a_j = (0, \ldots, 0). \tag{4.43}$$

4. *Setze*

$$x = \prod_{j \in S} x_j, \qquad \textit{und} \qquad y = \prod_{k=1}^{r} p_k^{d_k}, \quad d_k := \frac{1}{2} \sum_{j \in S} e_{jk}. \tag{4.44}$$

5. *Ist* $x \not\equiv_n \pm y$, *so liefert* $\mathrm{ggT}(x \pm y, n)$ *mindestens einen nichttrivialen Teiler von* n.

6. *Andernfalls starte mit einer neuen Faktorbasis und neuen* x_j.

Der Witz an (4.44) ist natürlich die Tatsache, daß $x^2 \equiv_n y^2$ ist. Dazu bemerken wir, daß S so gewählt ist, daß

$$\sum_{j \in S} (e_{j1}, \ldots, e_{jr}) \in (2\mathbb{Z})^r$$

und damit ist

$$y^2 = \prod_{k=1}^{r} p_k^{\sum_{j \in S} e_{jk}} = \prod_{k=1}^{r} \prod_{j \in S} p_k^{e_{jk}} = \prod_{j \in S} \underbrace{\prod_{k=1}^{r} p_k^{e_{jk}}}_{\equiv_n x_j^2} \equiv_n \left(\prod_{j \in S} x_j \right)^2 = x^2.$$

Interessant sind also eigentlich nur zwei Fragen: Wie findet man die Zahlen x_j in (4.42) und wie die Teilmenge S für (4.43). Die zweite Frage ist leichter zu beantworten, denn hier ist nur ein **homogenes lineare Gleichungssystem** der Form

$$0 = cA = (c_1, \ldots, c_s) \begin{bmatrix} a_1 \\ \vdots \\ a_s \end{bmatrix} = (c_1, \ldots, c_s) \begin{bmatrix} (e_{11})_2 & \ldots & (e_{1r})_2 \\ \vdots & \ddots & \vdots \\ (e_{s1})_2 & \ldots & (e_{sr})_2 \end{bmatrix}, \tag{4.45}$$

über dem Körper $\mathbb{F}_2$, das wegen $s > r$ **unterbestimmt** ist und daher immer nichttriviale Lösungen c hat, die sich beispielsweise durch **Gauß–Elimination**,

[129] In (Willems, 2008) heisst es „*. . . etwas größer, etwa* $r + 10$ *. . .*“

siehe (Sauer, 2013), bestimmen lassen. Die hierbei entstehende Mehrdeutigkeit der Lösung lässt sich auch ausnutzen, um eventuell dafür zu sorgen, daß $x \not\equiv_n \pm y$ ist und man so auch einen richtigen Teiler erhält. Jede Lösung c von (4.45) liefert dann sofort

$$S = \operatorname{supp} c = \{j \,:\, c_j \neq 0\}.$$

Bleibt also (4.42), und hier muss man ein wenig mehr Aufwand treiben, nämlich die als **quadratisches Sieb** bezeichnete Technik. Dazu betrachtet man das quadratische Polynom

$$f(x) = (x+m)^2 - n, \qquad m := \lfloor \sqrt{n} \rfloor, \tag{4.46}$$

und sucht s verschiedene Werte $x \in \{0, \pm 1, \pm 2, \dots\}$, für die $f(x)$ modulo n in die p_j faktorisierbar ist. Der Trick ist die folgende, wieder einmal recht einfache, Beobachtung.

Lemma 4.47 (Sieblemma) *Sei $p \in \mathbb{P}$. Dann gilt*

$$f(x) \equiv_p 0 \qquad \Leftrightarrow \qquad f(x + p\mathbb{Z}) \equiv_p 0, \tag{4.47}$$

und damit auch

$$f(x) = 0 \qquad \Rightarrow \qquad f(x + p\mathbb{Z}) \equiv_p 0, \tag{4.48}$$

Beweis: Die Richtung „$\Leftarrow$“ von (4.47) ist trivial, für die Umkehrung benötigen wir nur

$$\begin{aligned} f(x \pm p) &= (x \pm p + m)^2 - n \\ &= \underbrace{(x \pm p)^2}_{\equiv_p x^2} + 2\underbrace{(x \pm p)}_{\equiv_p x^2} m + m^2 - n \equiv_p x^2 + 2mx + m^2 - n \\ &= f(x) \equiv_p 0, \end{aligned}$$

und (4.48) folgt direkt aus (4.47). □

Mit dem f aus (4.46) wählt man nun $s \in \mathbb{N}$ und betrachtet das **Siebintervall** $\mathscr{S} = \{0, \pm 1, \dots, \pm s\}$. In $\mathscr{S}$ wollen wir die in F faktorisierbaren Werte $f(x)$ bestimmen, denn ist

$$\prod_{j=1}^{r} p_j^{e_j} = f(x) = (x+m)^2 - n \equiv_n (x+m)^2,$$

dann haben wir genau so ein x_j wie in (4.42) gefunden. Aber ganz $\mathscr{S}$ durchzuprobieren und jeden Wert $f(x)$ zu faktorisieren ist natürlich ein bisschen aufwendig.

Für einen clevereren Weg, $\mathscr{S}$ zu sieben, betrachtet man eine Primzahl $p \in F$ aus der Faktorbasis und löst

$$f(x) \equiv_p 0 \qquad \Leftrightarrow \qquad (x+m)^2 \equiv_p n, \qquad x \in \mathbb{Z}_p, \tag{4.49}$$

denn nur dann ist $f(x)$ für Faktorisierung durch p überhaupt interessant. Ist p klein genug, dann geht das einfach durch Ausprobieren, für größere p gibt es anspruchsvollere Methoden, in (Gathen & Gerhard, 1999) gibt es beispielsweise ein ganzes Kapitel zur Faktorisierung von Polynomen über endlichen

Körpern, was ja nun auch wieder äquivalent zum Nullstellenfinden ist. Die Gleichung (4.49) hat entweder keine Lösung, wenn n modulo p kein Quadrat ist, oder die beiden Lösungen $\pm(\sqrt{n})_p - m$, die wir mit x_{1p}, x_{2p} bezeichnen. Nach Lemma 4.47 gilt dann

$$x \in \mathscr{X}_p := (x_{1p} + p\mathbb{Z}) \cup (x_{2p} + p\mathbb{Z}) \qquad \Leftrightarrow \qquad f(x) \equiv_p 0,$$

das heisst, nur die Werte $f(x)$, $x \in \mathscr{S}_j = \mathscr{S} \cap \mathscr{X}_{p_j}$ können wirklich einen Faktor p_j^e enthalten, $j = 1, \ldots, r$.

Hat man also einmal die Mengen $\mathscr{S}_j$ ermittelt[130], dann kann man für $x \in \mathscr{S}' = \mathscr{S}_1 \cup \cdots \cup \mathscr{S}_r$ die Indemenge $R(x) = \{j \,:\, x \in \mathscr{S}_j\}$ ermitteln und mit

$$\mathrm{ggT}\left(\prod_{j \in R(x)} p_j^k, f(x)\right), \qquad k \in \mathbb{N},$$

so lange Potenzen der relevanten Elemente in F abdividieren bis der ggT den Wert 1 annimmt und auch keine weiteren Faktoren übrigbleiben. Der Vorteil des quadratischen Siebs liegt darin, daß man weiß, nach welchen Primzahlen man in welchen $f(x)$ zu suchen hat.

Es kann durchaus vorkommen, daß man für eine Wahl von $\mathscr{S}$ und F nicht genügend faktorisierbare Quadrate aussieben kann. In diesem Fall vergrößert man entweder das Siebintervall oder die Faktorbasis, wobei es natürlich jede Menge Möglichkeiten gibt, es gut oder schlecht zu machen.

Bemerkung 4.48 *Nach (Willems, 2008) ist die Dixon–Methode mit quadratischem Sieb momentan eine der schnellsten Faktorisierungsmethoden für ganze Zahlen.*

[130] Eine Nullstellenbestimmung für ein quadratisches Polynom und ein bisschen sieben.

Either mathematics is too big for the human mind or the human mind is more than a machine.

Kurt Gödel

5 Elliptische Kurven

Elliptische Kurven sind ein bisschen das „Nonplusultra“ der Kryptographie und sicherlich der komplexeste Ansatz, wobei es genau genommen ja um elliptische Kurven über endlichen Körpern $\mathbb{F}_p$, $p \in \mathbb{P}$, gehen wird. Tatsächlich erhält man so komplexere Gruppen, die gegen gewisse Angriffe resistenter sind.

5.1 Diskrete Logarithmen und ihre Schwachstellen

Als Motivation für die Einführung und Verwendung elliptischer Kurven betrachten wir eine Methode aus (Werner, 2002) zur schnellen Berechnung des diskreten Logarithmus in $\mathbb{F}_p$, die man als **Indexkalkül–Methode** bezeichnet. Gegeben ist wieder ein Erzeuger a der zyklischen Gruppe $\mathbb{F}_p^\times = \{a^k \ : \ k \in \mathbb{Z}_p\}$ sowie $x = a^y$ und gesucht ist $y = \log_a x$. Eigentlich suchen wir natürlich y nur modulo $p-1$, denn nach dem kleinen Fermat ist ja

$$a^{y+k(p-1)} = \underbrace{a^{k(p-1)}}_{\equiv_p 1}\, a^y \equiv_p a^y.$$

In diesem Sinne ist also $\log_a \ : \ \mathbb{F}_p \to \mathbb{Z}_{p-1}$.

Zur schnellen Bestimmung des diskreten Logarithmus bestimmt man in einem ersten Schritt die ersten t Primzahlen $P_t := \{p_1, \ldots, p_t\} \subset \mathbb{P}$ und versucht, deren diskrete Logarithmen zu bestimmen. Dazu betrachtet man zu einem normalerweise *zufälligen* $j \in \mathbb{Z}_p$ die Zahl a^j und versucht, diese mittels P_t zu faktorisieren. Klappt das, so ist

$$a^j = \prod_{k=1}^{t} p_k^{e_{jk}} \qquad \Leftrightarrow \qquad j \equiv_{p-1} \sum_{k=1}^{t} e_{jk} \log_a p_k.$$

Jedes „gute“ j liefert so eine lineare Gleichung und sobald die Matrix

$$E = \left[e_{jk} \ : \ \begin{matrix} j \in J \\ k = 1, \ldots, t \end{matrix} \right]$$

Rang t hat, dann gibt es eine invertierbare Teilmatrix, mit der wir $\log_a p_k$, $k = 1, \ldots, t$, durch Lösen eines *linearen Gleichungssystems* über $\mathbb{Z}_{p-1}$ bestimmen können. Hierfür gibt es effiziente Verfahren.

Hat man diese Vorberechnung einmal durchgeführt, so testet man, wieder für zufällige $j \in \mathbb{Z}_{p-1}$, ob $x\,a^j$ bezüglich P_t faktorisiert werden kann, also

$$a^{j+y} = x\,a^j \equiv_p \prod_{k=1}^{t} p_k^{m_k} \qquad \Rightarrow \qquad y \equiv_{p-1} -j + \sum_{k=1}^{t} m_k \log_a p_k$$

und die Werte auf der rechten Seite sind alle vorberechnet und bekannt.

Bemerkung 5.1 *Zur Rechnung in $\mathbb{Z}_{p-1}$ verwendet man idealerweise eine Primfaktorzerlegung von $p-1$ und den chinesischen Restsatz.*

Nach (Werner, 2002) kann dieses Verfahren mit subexponentieller Laufzeit aufwarten, was aus der relativ einfachen Struktur der multiplikativen Gruppe $\mathbb{F}_p^\times$ resultiert. Über elliptischen Kurven ist ein derartiger Ansatz nicht bekannt und es gibt wohl auch systematische Ursachen, die ihn unwahrscheinlich machen. Alles in allem ein guter Grund, sich mit elliptischen Kurven auseinanderzusetzen.

5.2 Definition elliptischer Kurven

Wir betrachten elliptische Kurven auf einem Körper $\mathbb{K}$ mit einer **Charakteristik**[131] verschieden von 2 und 3. Ausserdem wählen wir ein kubisches Polynom $f(x) = x^3 + ax + b$ in $\mathbb{K}[x]$.

Lemma 5.2 *Das Polynom $f(x) = x^3 + ax + b$ ist genau dann **quadratfrei**, d.h. es gibt kein q mit $q^2|f$, wenn $4a^3 + 27b^2 \neq 0$ ist.*

Beweis: Die Idee ist einfach: Ein Polynom f ist genau dann quadratfrei, wenn f und f' keine gemeinsame Nullstelle also keinen gemeinsamen Teiler haben, wenn also $\mathrm{ggT}(f, f') \in \mathbb{K}$ ist.

Nun ist $f'(x) = 3x^2 + a$ und der ggT der beiden Polynome berechnet sich nach dem euklidischen Algorithmus[132] über $\mathbb{R}$ als

$$\begin{array}{lll} x^3 + ax + b & 3x^2 + a & \to \frac{2}{3}ax + b \\ 3x^2 + a & \frac{2}{3}ax + b & \to a + \frac{27\,b^2}{4\,a^2} = \frac{4a^3+27b^2}{a^2} \end{array}$$

und der ggT ist $\neq 0$ wenn $4a^3 + 27b^2 \neq 0$ ist. □

Definition 5.3 (Elliptische Kurve) *Eine **elliptische Kurve** über $\mathbb{K}$ ist die Menge aller Punkte $(x, y) \in \mathbb{K}^2$ mit der Eigenschaft $y^2 = f(x)$, also*

$$E = E(f) = \left\{(x, y) \in \mathbb{K}^2 \;:\; y^2 = x^3 + ax + b\right\}, \tag{5.1}$$

*zu der noch ein **unendlich ferner Punkt O** dazukommt.*

[131] Die Charakteristik eines Körpers ist die kleinste Zahl n, für die die aus n Termen bestehende Summe $1 + \cdots + 1$ den Wert 0 annimmt; Körper wie $\mathbb{R}$, bei denen das nicht passiert, haben Charakteristik 0.

[132] Bei dem wir die Polynome vorher beliebig mit Einheiten, also von Null verschiedenen Zahlen, multiplizieren dürfen.

Bemerkung 5.4 *Offensichtlich ist* E *symmetrisch zur* x*–Achse, da* $(x, y) \in E$ *automatisch auch* $(x, -y) \in E$ *impliziert.*

Die Menge E ist eine **algebraische Kurve**, da sie Nullstellenmenge des Polynoms $F(x, y) = y^2 - x^3 - ax - b$ ist. Genau genommen ist das aber nur eine **affine Kurve**, da man sich im affinen Raum $\mathbb{K}^2$ bewegt, indem es eigentlich keinen affinen Punkt gibt. Der saubere Zugang ist es, eine **projektive Kurve** zu betrachten, bei der man ein **homogenes Polynom**

$$f(x) = \sum_{|\alpha|=n} f_\alpha x^\alpha, \qquad x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} x_3^{\alpha_3}$$

in $\mathbb{K}[x]$, $x = (x_1, x_2, x_3)$ betrachtet. Ein homogenes Polynom hat die Eigenschaft, daß

$$f(x) = 0 \qquad \Leftrightarrow \qquad f(\lambda x) = \lambda^n f(x) = 0, \quad \lambda \in \mathbb{K},$$

gilt und es daher mit den projektiven Äquivalenzklassen $[x] = \mathbb{K}\, x$ kompatibel ist. Im Projektiven gibt es nun die „normalen" Punkte x mit $x_3 \neq 0$ und die anderen, die unendlich fernen Punkte mit $x_3 = 0$, denn bekanntlich kann der affine Raum $\mathbb{K}^2$ mit $\mathbb{K}^1 \times \{1\}$ identifiziert werden. Das soll an dieser Stelle reichen, Details finden sich in (Werner, 2002), aber mittels projektiver Räume kann man den unendlich fernen Punkt tatsächlich sauber definieren.

Lemma 5.5 (Glattheit) *Eine elliptische Kurve ist* ***glatt****, d.h., für jeden Punkt* $(u, v) \in E$ *ist* $\nabla F \neq 0$.

Beweis: Da $F(x, y) = y^2 - f(x)$, ist

$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial F}{\partial x}(x, y) \\ \frac{\partial F}{\partial y}(x, y) \end{bmatrix} = \begin{bmatrix} f'(x) \\ 2y \end{bmatrix}$$

und $\nabla F(u, v) = 0$ bedeutet, daß $v = 0$ und $f'(u) = 0$ sein müssen. Da ausserdem $(u, v) \in E$ ist, ist außserdem $0 = v^2 = f(u)$ und damit wäre u eine doppelte Nullstelle von, ein Widerspruch zur Annahme, daß f quadratfrei ist. □

Von jetzt an bezeichnen wir Punkte auf der elliptischen Kurve mit $\mathbf{x} = (x_1, x_2)$, $\mathbf{y}, \mathbf{z}, \ldots$ und so weiter und definieren eine Gruppe auf E. Dazu beginnen wir mit der **Invertierung**

$$\mathbf{x} = (x_1, x_2) \qquad \Leftrightarrow \qquad -\mathbf{x} = (x_1, -x_2). \tag{5.2}$$

Um eine Addition $\mathbf{x} + \mathbf{y}$ zu definieren geht man nun geometrisch vor, siehe Abb 5.1. Die Gerade durch $\mathbf{x}$ und $\mathbf{y}$, im Bild P und Q schneidet die elliptische Kurve in einem dritten Punkt, der aber noch *nicht* die Summe ist, sondern deren (additives) Inverses $-\mathbf{x} - \mathbf{y}$. Ist $\mathbf{x} = \mathbf{y}$, berechnet man also sozusagen $2\mathbf{x}$, dann nehmen wir die Tangente an der Stelle $\mathbf{x}$, die nach Lemma 5.5 ja immer existieren muss.

Bemerkung 5.6 *Diese Konstruktion mag zuerst etwas seltsam erscheinen, allerdings ist „Addition auf kubischen Kurven" durchaus ein klassisches Gebiet mit der folgenden Intuition:*

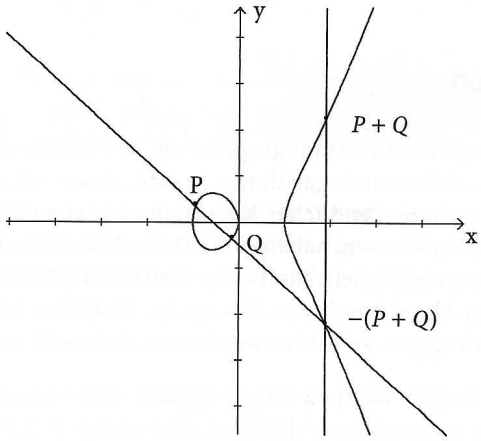


Abbildung 5.1: Geometrie der Addition von P und Q. Aus (Willems, 2008).

1. *Nach dem* ***Satz von Bézout***[133] *schneiden sich die kubische Kurve* F *und die Gerade* immer *in genau* $3 \cdot 1 = 3$ *Punkten, Vielfachheiten und unendlicher Punkt mitgezählt.*
2. *Diese drei Punkte nennen wir* $\mathbf{x}, \mathbf{y}, \mathbf{z}$.
3. *Ausserdem liegt auf der Geraden noch der unendlich ferne Punkt* $\mathbf{O}$, *denn eine Gerade hat ja unendliche Ausdehnung.*
4. *Diesen Zusammenhang schreibt man als* $\mathbf{x} + \mathbf{y} + \mathbf{z} = \mathbf{O}$ *oder eben* $-\mathbf{z} =: \mathbf{x} + \mathbf{y}$, *und genau das passiert in der Addition.*

Das alles ist natürlich kein *Beweis, noch nicht mal eine Heuristik, sondern nur eine intuitive Erklärung des Ganzen. Formal sauberer ist alles in (Werner, 2002) dargestellt.*

Über $\mathbb{R}$ kann man die Schnittpunkte und damit die Addition $\mathbf{z} = \mathbf{x} + \mathbf{y}$ sogar explizit berechnen. Wäre die Gerade durch $\mathbf{x}$ und $\mathbf{y}$ parallel zur y–Achse, dann wäre $\mathbf{y} = -\mathbf{x}$ und das Ergebnis wäre $\mathbf{O}$. Andernfalls ist $x_1 \neq y_1$ und die Gerade $y = \ell(x) = \alpha x + b$ kann explizit als

$$\ell(x) = x_2 \frac{x - y_1}{x_1 - y_1} + y_2 \frac{x_1 - x}{x_1 - y_1} = \frac{x_2 - y_2}{x_1 - y_1} x + \frac{x_1 y_2 - x_2 y_1}{x_1 - y_1} =: \alpha x + \beta$$

geschrieben werden. Sie nun $-\mathbf{z} = (z_1, -z_2)$ der dritte Schnittpunkt mit, dann ist

$$z_1^3 + a z_1 + b = z_2^2 = \ell(z_1)^2 = (\alpha z_1 + \beta)^2, \tag{5.3}$$

und x_1, y_1, z_1 sind die drei verschiedenen[134] Nullstellen des monischen kubischen Polynoms

$$q(x) = x^3 + ax + b - (\alpha x + \beta)^2 = x^3 - \alpha^2 x^2 + (a - 2\alpha\beta)x + b - \beta^2.$$

[133] Der Schnitt von algebraischen Kurven der Grade n und n' besteht aus genau $n\,n'$ Punkten. Das gilt allerdings nur über $\mathbb{C}$ und im projektiven Sinne

[134] Die Gerade ist ja gerade **nicht** vertikal!

Da

$$(x-\xi_1)(x-\xi_2)(x-\xi_3) = x^3 - (\xi_1+\xi_2+\xi_3)x^2 + \ldots,$$

folgt somit

$$\left(\frac{x_2-y_2}{x_1-y_1}\right)^2 = \alpha^2 = x_1+y_1+z_1$$

bzw.

$$z_1 = \left(\frac{x_2-y_2}{x_1-y_1}\right)^2 - x_1 - y_1. \tag{5.4}$$

Einsetzen in (5.3) liefert dann

$$z_2 = -\ell(z_1) = -\frac{x_2-y_2}{x_1-y_1}\,z_1 + \frac{x_2y_1-x_1y_2}{x_1-y_1} = -x_2 + \frac{y_2-x_2}{y_1-x_1}(x_1-z_1). \tag{5.5}$$

Damit ist $\mathbf{z} = \mathbf{x}+\mathbf{y}$ eindeutig definiert und sogar *ohne die explizite Kenntnis* von a, b berechenbar. Im Falle von $\mathbf{x} = \mathbf{y}$ funktioniert die obige Argumentation nicht mehr, sondern man muß als $\ell(x)$ die Tangente verwenden und kommt dann auf die Regel[135]

$$z_1 = \left(\frac{3x_1^2+a}{2x_2}\right)^2 - 2x_1, \qquad z_2 = -x_2 + \frac{3x_1^2+a}{2x_2}(x_1-z_1). \tag{5.6}$$

Diese Operationen verpassen der elliptischen Kurve dann eine Gruppenstruktur.

Satz 5.7 (Elliptische Kurve als Gruppe) *Mit den Rechenoperationen* (5.2), (5.4), (5.5) *und* (5.6) *wird* E *zu einer additiven abelschen Gruppe mit neutralem Element* $\mathbf{O}$.

Beweis: Eigentlich haben wir alles beisammen. Kommutativität ist klar, weil die Reihenfolge der Punkte, durch die die Gerade gelegt wird, irrelevant ist, die Inverse ebenfalls und die Assoziativität kann man explizit nachrechnen[136]. □

Bemerkung 5.8 *Da eine elliptische Kurve eine* ***additive Gruppe*** *ist, nimmt der diskrete Logarithmus nicht mehr die Form* a^k *an, sondern die Form* $k\mathbf{x}$ *und man muss jetzt aus gegebenem* $\mathbf{x}$ *und* $\mathbf{y} = k\mathbf{x}$ *den Faktor* k *errechnen.*

Man sollte sich klarmachen, daß die Additionsformeln für die elliptische Kurve in *jedem Körper* funktionieren, nicht nur über $\mathbb{R}$, denn alles was wir tun ist addieren, subtrahieren und dividieren[137]. Der einzige Grund, warum wir $\mathbb{R}$ verwendet haben, war die **geometrische Interpretation** der Gruppenoperation. Ist $\mathbb{K}$ nun ein endlicher Körper, also beispielsweise $\mathbb{F}_p$, $p \in \mathbb{P}$, dann ist E eine **endliche abelsche Gruppe**, allerdings jetzt additiv geschrieben. Sehen wir uns einmal ein Beispiel an.

[135] In der man sehr deutlich den Gradienten erkennen kann.

[136] Oder, um es mit (Gathen & Gerhard, 1999) zu sagen: *„The latter is not hard to check on a Computer Algebra system"*.

[137] Aber eben nicht durch 0.

Beispiel 5.9 (Elliptische Kurve) *Sei* $f(x) = x^3 - 2x$ *über* $\mathbb{F}_5$*, wir betrachten also die Lösungen von* $y^2 = x^3 - 2x$*. Die Punkte auf der Kurve sind*

$$\begin{array}{lll} x = 0 & y = 0 & \Rightarrow y = 0 \\ x = 1 & y^2 = -1 \equiv_5 4 & \Rightarrow y = \pm 2 \\ x = 2 & y^2 = 4 & \Rightarrow y = \pm 2 \\ x = 3 & y^2 = 21 \equiv_5 1 & \Rightarrow y = \pm 1 \\ x = 4 & y^2 = 56 \equiv_5 1 & \Rightarrow y = \pm 1, \end{array}$$

die elliptische Kurve ist also die 10*–elementige abelsche Gruppe*

$$E = \{(0,0), (1,\pm 2), (2,\pm 2), (3,\pm 1), (4,\pm 1)\} \cup \mathbf{O},$$

wobei $\mathbf{O}$ *nun ein rein formales Objekt ist. Setzen wir* $\mathbf{x} = (1,2)$ *und* $\mathbf{y} = (3,1)$*, dann erhalten wir für* $\mathbf{z} = \mathbf{x} + \mathbf{y}$*, daß*[138]

$$z_1 = \left(\frac{2-1}{1-3}\right)^2 - 1 - 3 = (-1\cdot 3)^2 - 4 = 9 - 4 = 5 \equiv_5 0$$

und

$$z_2 = -2 + \frac{1-2}{3-1}(1-0) = -2 - 1\cdot 3 = -5 \equiv_5 0,$$

also $\mathbf{z} = (0,0)$*.* ***Achtung:*** $(0,0)$ *ist* ***nicht*** *das neutrale Element von* E*, das ist* $\mathbf{O}$*!*

Ganz analog rechnet man mit (5.6) *nach, daß*

$$2\mathbf{x} = (4,1), \qquad 4\mathbf{x} = 2(2\mathbf{x}) = (2,-1), \qquad 5\mathbf{x} = \mathbf{x} + 4\mathbf{x} = (4-1),$$

weswegen $\mathbf{x}$ *weder die Ordnung* 2 *noch die Ordnung* 5 *hat. Da* $\#E = 10$ *ist und die Ordnung von* $\mathbf{x}$ *die Kardinalität der Gruppe teilen muss, kann die Ordnung von* $\mathbf{x}$ *nur* 10 *sein und* E *ist eine* ***zyklische Gruppe****:* $E = \{j\mathbf{x} \,:\, j \in \mathbb{Z}_{10}\}$*.*

Übung 5.1 Implementieren Sie die Rechenoperation für die elliptische Kurve aus obigem Beispiel. ◊

Das Ziel aus kryptographischer Hinsicht ist es nun, elliptische Kurven mit vielen Elementen und von großer Ordnung zu konstruieren, die noch dazu zyklisch sind oder zumindest Elemente großer Ordnung enthalten und in denen man dann einen diskreten Logarithmus verwenden kann. Aber vorher noch eine Warnung.

Bemerkung 5.10 (Side Channel Attack) *Das Rechnen in elliptischen Gruppen ist komplex und viele der Verfahren, die wir in* $\mathbb{Z}_p$ *kennengelernt haben und die auf dem kleinen Fermat beruhen, funktionieren auf elliptischen Kurven nicht mehr – schließlich gibt es keine Teilbarkeit und Primelemente und dergleichen. Allerdings muss man bei der Implementierung sehr vorsichtig sein, denn die Additionsformeln* (5.4)/ (5.5) *und* (5.6) *sind unterschiedlich und sorgen auch für ein unterschiedliches Verhalten des Rechners, sei es in Sachen Stromverbrauch, sei es akustisch. Und diese Information könnte ein Angreifer abhören und zum Bestimmen des diskreten Logarithmus verwenden.*

[138] Hinweis: In $\mathbb{F}_5$ ist $2^{-1} = 3$.

Ein Ausweg bestünde darin, verschiedene *Implementierungen der Rechenoperationen zu haben, die gegebenenfalls zufällig aufgerufen haben und ein möglichst identisches Verhalten aufweisen. Das zeigt, daß Krptographie letztendlich auch sehr sorgfältig implementiert werden muss.*

Wir betrachten von nun an elliptische Kurven über dem endlichen Körper $\mathbb{F}_p$, $p \in \mathbb{P}$, und zwar als additive Gruppen. Die erste, einfache Frage ist natürlich die Größe dieser Gruppe.

Lemma 5.11 $\#E \leq 2p + 1$.

Beweis: Für jeden der p Punkte $x \in \mathbb{F}_p$ gibt es die maximal zwei Elemente der elliptischen Kurve, nämlich $\left(x, \pm(\sqrt{f(x)})_p\right)$, und dann kommt noch $\mathbf{O}$ dazu. □

Das ist natürlich eine ziemlich primitive *ad hoc*–Abschätzung, die man sicherlich verbessern kann und auch sollte. Eine schöne Heuristik dazu findet sich wieder einmal in (Gathen & Gerhard, 1999). Es basiert auf der einfachen Idee, daß ein Punkt (x, y) genau dann auf der elliptischen Kurve liegt, wenn $f(x) \in \mathbb{F}_p^2$ liegt, also ein **Quadrat** ist. Wie wahrscheinlich ist das? Schauen wir uns die Gruppe $\mathbb{F}_p^\times = \{a^j \,:\, j \in \mathbb{Z}_{p-1}\}$ an, dann ist $x = a^j$ genau dann ein Quadrat, wenn j gerade ist, also bei $j = 0, 2, \ldots, p-3$, für $j = 1, 3, \ldots, p-2$ ist es kein Qudrat. Fazit, bei Annahme von Gleichverteilung auf $\mathbb{F}_p^\times$ gilt

$$P\left(x \in (\mathbb{F}_p^\times)^2\right) = \frac{1}{2}.$$

Und dann gibt's halt noch die 0, die eine Sonderrolle spielt, denn sie ist ein Quadrat, aber mit der Eigenschaft, daß $\pm\sqrt{0} = 0$. Das kann man auch sehr schön mit dem **Legendre–Symbol** aus (4.16) hinschreiben:

$$\sum_{x \in \mathbb{F}_p} \left(\frac{x}{p}\right) = 0, \tag{5.7}$$

und für die Kardinaliät der elliptischen Kurve gilt die schöne Formel

$$\#E = 1 + \sum_{x \in \mathbb{F}_p} \left(1 + \left(\frac{f(x)}{p}\right)\right) = 1 + p + \sum_{x \in \mathbb{F}_p} \left(\frac{f(x)}{p}\right), \tag{5.8}$$

die uns so richtig aber auch nicht weiterhilft, denn die magische Summe über das Legendre–Symbol von $f(x)$ ist ja unbekannt. Wäre f nun eine Bijektion von $\mathbb{F}_p$ nach $\mathbb{F}_p$, dann wäre $\#E = 1 + p$, das wäre so ziemlich die kleinste elliptische Kurve über $\mathbb{F}_p$. Die Frage ist also: Wie weit kann diese Summe über Legendre–Polynome von der Gleichverteilung abweichen und die Antwort darauf ist der folgende ziemlich tiefliegende Satz[139].

[139] In keinem der Bücher, die ich für die Ausarbeitung der Vorlesung benutzt habe, wurde er bewiesen, sondern immer nur zitiert, in (Gathen & Gerhard, 1999) wenigstens motiviert. Also machen wir es hier auch so.

Satz 5.12 (Hasse–Schranke, (Hasse, 1933)) *Für eine elliptische Kurve E über dem Körper $\mathbf{F}_p$ gilt*

$$|\#E - (p+1)| \leq 2\sqrt{p}. \tag{5.9}$$

Wenn wir also eine elliptische Kurve mit einem Element hoher Ordnung und damit auch mit vielen Elementen wollen, dann bleibt uns gar nichts anderes übrig, als auch einen Körper hinreichender Größe zu verwenden. Man sollte allerdings dazusagen, daß elliptische Kurven für *alle* Körper funktionieren, also auch für $\mathbb{F}_q$, $q = p^k$, wie aus Proposition 2.34.

Übung 5.2 Schreiben Sie ein `Matlab`–Programm, das für kleine $p \in \mathbb{P}$, also z.B. $\log_2 p < 32$, und vorgegebene $a, b \in \mathbb{F}_p$ die elliptische Kurve E bestimmt und die Rechenregeln auf E umsetzt.
Hinweis: Dafür braucht man auch eine Implementierung des erweiterten euklidischen Algorithmus 2.22. ◊

Bemerkung 5.13 *Es gibt eine Verfahren, mit dem man sehr effizient die Anzahl der Elemente einer elliptischen Kurve berechnen kann, nämlich den* ***Schoof–Algorithmus****, (Schoof, 1985). In dieser Arbeit gibt es auch einen effizienten Algorithmus zum Auffinden der Wurzel $(\sqrt{n})_p$, vorausgesetzt n ist eine Quadratzahl. Beide Verfahren gehen aber weit über den Umfang dieser Vorlesung hinaus*[140]*, da sie substantielle Theorie über elliptische Kurven benötigen.*

Beispiel 5.14 (Struktur elliptischer Kurven) *Über $\mathbb{F}_7$ bestehen die elliptischen Kurven $E_\pm$ zu $f(x) = x^3 - x$ und $f(x) = x^3 + x$ jeweils aus 8 Punkten, und zwar*

$$E_- = \{(0,0), (1,0), (4,2), (4,5), (5,1), (5,6), (6,0), \mathbf{O}\}$$

und

$$E_+ = \{(0,0), (1,3), (1,4), (3,3), (3,4), (5,2), (5,5), \mathbf{O}\},$$

aber strukturell sind die Gruppen verschieden, denn E_- wird von den beiden Elementen $(4,2)$ (Ordnung 4) und $(0,0)$ (Ordnung 2) erzeugt und erfüllt daher $E_- \simeq \mathbb{Z}_4 \times \mathbb{Z}_2$, wohingegen in E_+ beispielsweise $(3,3)$ die Ordnung 8 hat und somit die ***zyklische Gruppe*** *erzeugt. Das heißt natürlich insbesondere $E_+ \simeq \mathbb{Z}_8$.*

Es gibt übrigens laut (Gathen & Gerhard, 1999, Table 19.7) auch wirklich eine elliptische Kurve über $\mathbb{F}_7$ mit der Maximalzahl von 7 Elementen.

5.3 Faktorisieren mit elliptischen Kurven

Einen großen Vorteil elliptischer Kurven kennen wir schon, nämlich das sehr „unstrukturierte“ Verhalten der Rechenoperationen und das weitgehende Fehlen von Struktur in der Gruppe. Man kann elliptische Kurven aber auch benutzen[141], um Zahlen zu faktorisieren, was zu **Lenstras Methode** führt. Wir starten wie gehabt mit einer Zahl n, die wir faktorisieren wollen.

[140] Im Vergleich dazu ist der Beweis des AKS–Verfahrens ein Kinderspiel.
[141] Sollte man sagen „missbrauchen“?

Im ersten Schritt bestimmt man eine zufällige elliptische Kurve E und einen Punkt $\mathbf{x} \in E$, indem man x_1, x_2, a *zufällig* in $\mathbb{Z}_n$ wählt,

$$b := x_2^2 - x_1^3 - ax_1$$

setzt und testet ob die so bestimmte Kurve F mit $F(\mathbf{x}) = 0$ modulo n auch qudratfrei ist, indem man[142]

$$d := \mathrm{ggT}\,(4a^3 + 27b^2, n)$$

berechnet. Ist $d = 1$, ist die Kurve qudratfrei, ist $1 < d < n$, dann haben wir einen Teiler von n gefunden und sind fertig und ist $d = n$, dann verwerfen wir die Kurve und beginnen von neuem.

Wir können also nun annehmen, daß wir eine elliptische Kurve mit $f(x) = x^3+ax+b$ und einen Punkt $\mathbf{x} \in \mathbb{Z}_n^2$ haben. Nun tun wir so, als wäre $\mathbb{Z}_n$ ein Körper und berechnen vermittels (5.6) und (5.4)/ (5.5) die Punkte $2\mathbf{x}, 3\mathbf{x}, \ldots$ Wenn diese Berechnung scheitert, dann liegt es daran, daß wir durch eine Zahl in $y \in \mathbb{Z}_n \setminus \mathbb{Z}_n^\times$ dividieren wollten, die nach Proposition 2.28 dann $\mathrm{ggT}\,(k, n) \neq 1$ erfüllen muss, und schon wieder haben wir einen Teiler von n gefunden.

Jetzt sei also $\mathbb{P} \ni p \leq \sqrt{n}$, $p|n$, ein Primteiler von n, und wir nehmen an, daß wir all die Punkte $\mathbf{x}, 2\mathbf{x}, \ldots, k\mathbf{x}$ für ein $k \geq 2$ bereits berechnet haben. Nun seien

$$\overline{a} := (a)_p = a - sp, \qquad \overline{b} := (b)_p = b - tp$$

und da $\overline{a}^3 \equiv_p a^3$ sowie $\overline{b}^3 \equiv_p b^3$, liefert uns

$$\mathrm{ggT}\,(4\overline{a}^3 + 27\overline{b}^2, p) \,|\, \mathrm{ggT}\,(4a^3 + 27b^2, n) = 1,$$

daß das Polynom $\overline{f}(x) = x^3 + \overline{a}x + \overline{b}$ über $\mathbb{F}_p$ quadratfrei ist[143], also ist

$$\overline{E} = \left\{ \mathbf{y} \in \mathbb{F}_p^2 \;:\; y_2^2 = \overline{f}(y_1) \right\} \subseteq \mathbb{F}_p^2$$

eine elliptische Kurve über $\mathbb{F}_p$ und $\mathbf{y} = (k\mathbf{x})_p$ ein Punkt auf dieser Kurve.

Lemma 5.15 *Für zwei Punkte $\mathbf{x}, \mathbf{y} \in E$ über $\mathbb{F}_p$ folgt aus $\mathbf{x} + \mathbf{y} = \mathbf{O}$ daß entweder $\mathbf{x} \neq \mathbf{y}$ und $x_1 = y_1$ oder $\mathbf{x} = \mathbf{y}$ und $x_2 = 0$ ist.*

Beweis: $\mathbf{x}+\mathbf{y} = \mathbf{O}$ bedeutet $(x_1, x_2) = -\mathbf{y} = (y_1, -y_2)$, also $x_1 = y_1$ und $x_2 = -y_2$. Die erste dieser Identitäten liefert $x_1 = y_1$ und falls $\mathbf{x} = \mathbf{y}$ ist, so ergibt die zweite, daß $x_2 = -x_2 = 0$ sein muss. □

Da wir $k\mathbf{x}$ nur dann erreichen können, wenn wir nicht vorher durch Null dividieren, ist nach Lemma 5.15 $k\mathbf{x} \neq \mathbf{O}$ und damit auch $\mathbf{y} \neq \mathbf{O}$. Da also $\mathbf{y} = (j\mathbf{x})_p = j(\mathbf{x})_p$ nicht $\mathbf{O}$ ist, $j = 1, \ldots, k$, enthält die $\overline{E}$ einen Punkt der Ordnung k. Da andererseits, nach dem Satz von Hasse

$$\#\overline{E} \leq p + 1 + \sqrt{p} \leq \sqrt{n} + \sqrt[4]{n}$$

[142] Natürlich nur modulo n.

[143] Zur Erinnerung: Qudratfreiheit hängt vom Körper ab! Beispielsweise ist $f(x) = x^3 - x = x(x+1)(x-1)$ über $\mathbb{F}_2$ nicht quadratfrei, über $\mathbb{F}_3$ schon.

ist, muss nach spätestens $O(\sqrt{n})$ Schritten bei der Division etwas schiefgehen und wir haben einen nichttrivialen Faktor von n gefunden.

Eine Verbesserung dieses Verfahrens, die nicht addiert, sondern mit iteriertem Verdoppeln arbeitet[144], ist in (Gathen & Gerhard, 1999, Algorithm 19.22) beschrieben.

Bemerkung 5.16 (Lenstra)

1. *Das Verfahren funktioniert dann besonders gut, wenn $\overline{E}$ eine möglichst kleine Ordnung hat, denn dann muss schon bei sehr kleinen Werten von k eine „Division durch Null" auftreten. Das passiert dann, wenn $\#\overline{E}$ nur kleine Primteiler hat, denn dann gibt es viele Punkte $\mathbf{x} \in \overline{E}$ mit $q\mathbf{x} = \mathbf{O}$, wobei q einer der Primteiler von $\#\overline{E}$ ist.*

2. *Diese „gute" Eigenschaft hängt vom Körper und von dem gewählten Polynom f, also von a und b, ab, siehe Beispiel 5.14, wenn man es genau nimmt aber natürlich auch von dem **unbekannten** Primfaktor p.*

3. *Trotzdem erhält man so mit der Wahl der elliptischen Kurve eine zusätzliche Flexibilität.*

4. *Laut (Willems, 2008) ist Lenstras Methode besonders gut, wenn $n = pq$ mit $p \sim q \sim \sqrt{n}$.*

Da wir ja die drei Parameter der a, x_1, x_2 zur Bestimmung der elliptischen Kurve über $\mathbb{Z}_n$ ja vorhin *zufällig* gewählt haben, stellt sich die Frage, wie hoch die „Trefferquote" eigentlich ist. Dazu noch einen schönen Begriff.

Definition 5.17 *Für $a, b \in \mathbb{N}$ heißen die Elemente der Menge*

$$\Psi(a, b) = \{1 \le k \le a \ : \ \mathbb{P} \ni p | a \ \Rightarrow \ p \le b\}$$

*b–**glatte Zahlen** kleiner oder gleich a. Dies sind also all Zahlen, die nur Primfaktoren kleiner oder gleich b haben. Außerdem setzen wir $\psi(a, b) := \#\Psi(a, b)$.*

Rechnen wir doch spaßeshalber mal aus, wie groß dieses $\psi(a, b)$ so werden kann, wobei natürlich im Normalfall $a \gg b$ sein sollte, $a \ge b$ aber vorauszusetzen ist. Dazu setzt man $u := \frac{\log a}{\log b} \ge 1$, $v = \lfloor u \rfloor$, und stellt fest, daß

$$b = e^{\log b} = e^{\log a (\log a)^{-1} \log b} = a^{(\log a)^{-1} \log b} = a^{1/u} \qquad \Leftrightarrow \qquad b^u = a$$

sein muss. Mit $\mathbb{P}_{\le k} := \mathbb{P} \cap \mathbb{Z}_{k+1}$ betrachten wir nun einen Vektor $(q_1, \ldots, q_v) \in \mathbb{P}_{\le b}$ und setzen $q = q_1 \cdots q_v$. Dann ist $q \le b^v \le b^u = a$ und daher $q \in \Psi(a, b)$. Für jedes solche q gibt es maximal $v!$ Möglichkeiten, es als $q_1 \cdots q_v$

[144] So wie das iterierte Quadrieren bei der Exponentiation.

zu schreiben und da $\#\mathbb{P}_{\leq b} = \pi(b)$ ist, liefert uns die quantitative Version (4.34) des Primzahlsatzes[145], daß

$$\begin{aligned}\psi(a,b) &\geq \frac{\pi(b)}{\nu!} \geq \left(\frac{\pi(b)}{\nu}\right)^{\nu} \geq \left(\frac{b}{\nu \log b}\right)^{\nu} = \left(\frac{b}{\log b}\right)^{\nu} \approx \frac{b^u}{(\log b)^u} u^{-u} \\ &= x\,(u \log b)^{-u}\end{aligned}$$

sein muss, zumindest, wenn u nicht zu groß wird, also $\log b \sim \log a$ ist. Das kann man über **Dickmanns ρ–Funktion** $u \mapsto u^{-u}$ auch als

$$\frac{\psi(x, x^{1/u})}{x} = u^{-u(1+o(1))} \tag{5.10}$$

schreiben. Es gibt also tatsächlich b–glatte Zahlen und für $x \leq a$ gilt dann

$$P\left(x \in \Psi(a,b)\right) = \frac{\psi(a,b)}{a} \geq (u \log b)^{-u} = (u \log a^{1/u})^{-u} = (\log a)^{-u}.$$

Auf der Basis dieser Größe, die auch bei einer quantitativen Analyse der Dixon–Methode, Algorithmus 4.46 benötigt wird, gibt es dann eine Abschätzung für die Wahrscheinlichkeit „guter“ elliptischer Kurven, siehe (Gathen & Gerhard, 1999, Corollary 19.25).

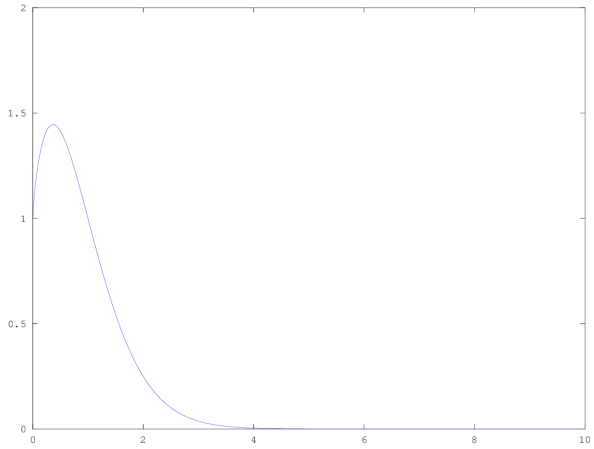


Abbildung 5.2: Die Funktion $u \mapsto u^{-u}$, falls sie jemanden interessiert. Man sieht eigentlich nur, daß sie recht schnell abfällt.

Proposition 5.18 *Sei im Lenstra–Verfahren p ein Primteiler von n und b ein technischer Parameter im Verfahren*[146]*. Dann gibt es eine positive Konstante C so daß für die Anzahl m der Tripel a, x_1, x_2, für die die Faktorisierung erfolgreich ist, die Abschätzung*

$$\frac{m}{n^3} \geq C \frac{\psi(p+1-\sqrt{p}, b) - \psi(p+1+\sqrt{p}, b)}{\sqrt{p}\,\log p} \tag{5.11}$$

[145] Zumindest für $b > 59$.

[146] Das k in kx wird in Primzahlen faktorisiert und b ist eine obere Schranke für diese Primzahlen.

erfüllt ist.

Damit wollen wir es erst einmal gut sein lassen mit den elliptischen Kurven und uns dem diskreten Logarithmus und dessen Berechnung zuwenden.

Hätte er eine Flinte gehabt und ein Kornfeld: Er würde erstere in letzteres geworfen haben.

Gustav Meyrink, *Walpurgisnacht*

6 Diskrete Logarithmen

In diesem Kapitel wenden wir uns nun effizienten Methoden zur Berechnung des diskreten Logarithmus zu, also der Bestimmung von x, so daß zu gegebenem y, a im multiplikativen Fall $y = a^x$, im additiven[147] Fall $y = x\,a$ erfüllt ist.

Wir wollen uns hier auf eine *multiplikative* Schreibweise der Gruppe beschränken, damit $\log_a x$ auch mit der Intuition übereinstimmt. Legen wir also erst einmal das „Setup“ fest.

Definition 6.1 (Diskreter Logarithmus in Gruppen) *Es sei G eine multiplikative Gruppe, $a \in G$ ein Element der **Ordnung** n und*

$$G_a := \langle a \rangle := \{a^k \,:\, k \in \mathbb{Z}_n\} \subseteq G.$$

*Die **Logarithmusfunktion** $\log_a : \langle a \rangle \to \mathbb{Z}_n$ bestimmt zu $y \in G_a$ das eindeutige Element $x \in \mathbb{Z}_n$ mit $y = a^x$.*

Zur Berechnung von $\log_a$ müssen wir zwischen zwei Typen von Algorithmen unterscheiden: Solchen, die für *jede* Gruppe, also die ganz allgemeine Situation von Definition 6.1 funktionieren, und solche, die eine spezielle Struktur von G bzw. G_a ausnutzen wie die **Indexkalkül–Methode** aus Abschnitt 5.1.

Die einfachste Technik zur Bestimmung des diskreten Logarithmus ist natürlich ausprobieren, also fröhliches Exponentieren von a, was eine Laufzeit von $O(n)$ mit sich bringt[148], dafür aber immer funktioniert. Sehen wir uns ein paar Verfahren an, die zumindest ein besseres Laufzeitverhalten bieten. Dabei ist jetzt immer $y \in \langle a \rangle$ gegeben und $x = \log_a y$ gesucht.

6.1 Allgemeine Verfahren

Wir beginnen mit einem einfachen Verfahren, das auf Kosten des Speichers zu einer besseren Laufzeit kommt als reines Ausprobieren. Dieses Verfahren wird als **Babystep–Giantstep** bezeichnet und versucht, ein paar Potenzen hoher und ein paar niedriger Ordnung zu verwenden. Dazu setzen wir $m := \lceil n \rceil$.

[147] Wie bei den elliptischen Kurven.

[148] n steht jetzt immer für die **Ordnung** von a.

Baby Step: Bestimme die Menge

$$B := \{(y\,a^{-r}, r) \;:\; r \in \mathbb{Z}_m\} \subset \langle a\rangle \times \mathbb{Z}_m. \tag{6.1}$$

Enthält B ein Element der Form $(1, r)$, dann ist $y = a^r$, also $x = \log_a y = r$ und das Problem ist bereits gelöst.

Giant Step: Für $j = 1, \ldots, m-1$ suchen wir in B nach einem Paar (b, r) mit

$$y a^{-r} = b = a^{mj}. \tag{6.2}$$

Ein Paar wie in (6.2) muss es immer irgendwann geben, da wir per Division mit Rest x als $x = jm + r$ für passende $j \le m-1$, $r \in \mathbb{Z}_m$ schreiben können und dann

$$y = a^x = a^{jm+r} \qquad \Leftrightarrow \qquad y a^{-r} = a^{mj}$$

sein muss. Die Laufzeit des Algorithmus ist von der Größenordnung $O(\sqrt{n})$, also immer noch exponentiell in der Länge von n, wobei beim *Giant Step* relativ wenig zu rechnen und viel zu suchen ist.

Ein bisschen cleverer ist dann schon der **Pohlig–Hellman–Algorithmus**, der mit einer Primfaktorisierung der Ordnung n von a startet:

$$n = \prod_{j=1}^{r} p_j^{e_j}, \qquad p_j \in \mathbb{P}, \quad e_j \in \mathbb{N}. \tag{6.3}$$

Diese Ordnung muss bekannt sein, aber sie muss nur ein einziges Mal bestimmt werden, ebenso wie ihre Primfaktorzerlegung. Der Trick des Verfahrens besteht darin, den diskreten Logarithmus x nicht direkt zu bestimmen, sondern Zahlen $x_j \in \mathbb{Z}_{p_j^{e_j}}$, $j = 1, \ldots, r$, mit der Eigenschaft

$$x_j \equiv_{q_j} x, \qquad q_j := p_j^{e^j}, \qquad j = 1, \ldots, r, \tag{6.4}$$

zu bestimmen und aus diesen x über den chinesischen Restsatz, Algorithmus 2.36, zu bestimmen.

Lemma 6.2 *Jeder Zahl $x \in \mathbb{Z}_{p^k}$, $p \in \mathbb{P}$, kann auf eindeutige Art und Weise als*

$$x = \sum_{j=0}^{k-1} a_j\, p^j, \qquad a_j \in \mathbb{Z}_p, \tag{6.5}$$

geschrieben werden.

Beweis: Das sieht schlimmer aus als es ist, denn wenn man einmal vergisst, daß p eine Primzahl ist, dann ist dies nichts als die p–adische Darstellung der Zahl $0 \le x < p^k$, also die Darstellung zur **Basis** p. □

Das gilt dann natürlich auch und gerade für die Darstellung

$$x_j = \sum_{k=0}^{e_j-1} a_{jk}\, p_j^k, \qquad a_{jk} \in \mathbb{Z}_{p_j},$$

deren Koeffizienten man nun wie folgt bestimmt:

1. Für den Koeffizienten a_{j0} definiert man das Element $b := a^{n/p_j}$ von der Ordnung p_j und bemerkt, daß

$$y^{\frac{n}{p_j}} = a^{x\frac{n}{p_j}} = b^x = b^{a_{j0}}.$$

   Damit können wir a_{j0} dadurch finden, daß wir den Wert $y^{\frac{n}{p_j}}$ in der (kleinen) Menge $\{b^k \,:\, k \in \mathbb{Z}_{p_j}\}$ suchen.

2. Nehmen wir nun an, wir hätten $a_{j0}, \ldots, a_{j,k-1}$ schon bestimmt und suchen a_{jk}, $k \geq 1$. Dazu setzen wir

$$y_k := y\, a^{-a_{j0}-a_{j1}p_j-\cdots-a_{j,k-1}p_j^{k-1}} = y^{a_{jk}p_j^k+a_{j,k+1}p_j^{k+1}+\cdots} =: y^{a_{jk}\,p_j^k}\, y^{d\,p_j^{k+1}}$$

   und nutzen

$$y_k^{n/p_j^{k+1}} = a^{a_{jk}\frac{n}{p_j}} a^{dn} = b^{a_{jk}}\, b^{dp_j} = b^{a_{jk}},$$

   um wie gewünscht a_{jk} zu bestimmen.

Das sollte man noch einmal in einem Algorithmus zusammenfassen.

Algorithmus 6.3 (Pohlig–Hellman–Algorithmus)
Gegeben: $a \in G$ *von der Ordnung* n, $y \in G$.

1. *Bestimme die Primfaktorzerlegung*

$$n = \prod_{j=1}^{r} p_j^{e_j}$$

   von n.

2. *Berechne die Werte* $b_j = a^{n/p_j}$ *und die Mengen*

$$B_j = \{b_j^k \,:\, k \in \mathbb{Z}_{p_j}\}, \qquad j = 1, \ldots, r.$$

3. *Für* $j = 1, \ldots, r$

   (a) *Setze* $z_0 = 1$.

   (b) *Für* $k = 0, \ldots, e_j - 1$

      i. *Berechne*

$$y_{jk} = (yz_k)^{\frac{n}{p_j^{k+1}}}$$

      und bestimme a_{jk} *als Index von* y_{jk} *in* B_j.

      ii. *Setze* $z_{k+1} = z_k\, a^{-a_{jk}p_j^k}$.

   (c) *Setze*

$$x_j = \sum_{k=0}^{e_j} a_{jk}\, p_j^k.$$

4. *Bestimme* x *aus den Kongruenzen* $x \equiv_{p_j^{e_j}} x_j$ *durch den chinesischen Restsatz.*

Ergebnis: $x = \log_a y$.

Bemerkung 6.4

1. *Die Schritte 1) und 2 in Algorithmus 6.3 hängen nur von* n*, aber nicht von* y *und noch nicht einmal von* a*. Will man also* $\log_a$ *für verschiedene Werte von* a *berechnen, so muss man sich die Arbeit nur einmal machen.*

2. *Die Berechnung der* x_j *ist perfekt parallelisierbar.*

3. *Schlechte Gruppenordnungen für den diskreten Logarithmus haben also viele kleine Primfaktoren mit nicht zu hohem Exponenten, wobei der Exponent nur logarithmisch, nämlich über die Anzahl der* p*–adischen Stellen von* x_j *eingeht.*

4. *Der Algorithmus funktioniert auf* jeder *Gruppe, denn faktorisiert wird in multiplikativer Schreibweise nur der Exponent und der ist eine ganze Zahl.*

Es gibt noch mehr Verfahren zur Berechnung des diskreten Logarithmus, unter anderem auch eine Variation der Pollardschen ρ–Methode oder eine Pollardsche λ–Methode. Angegeben sind diese in (Werner, 2002).

6.2 Ein spezielles Verfahren für elliptische Kurven

Es gibt auch Verfahren, die speziell auf die Bestimmung des diskreten Logarithmus in elliptischen Kurven zugeschnitten sind. Im vollen Detail sind diese Methoden sehr anspruchvoll und weit jenseits von dem, was man in dieser Vorlesung behandeln kann. Allerdings geben sie auch wichtige Informationen, wie man[149] „gute“ und „schlechte“ elliptische Kurven unterscheiden kann.

Für das sogenannte **MOV–Verfahren** benötigen wir das Konzept „**Abschluss eines Körpers**“, das im wesentlichen aus dem Körper und allen Wurzeln, also allen Nullstellen von Polynomen über dem Körper besteht. Bei endlichen Köpern ist das im wesentlichen die Konstruktion aus Proposition 2.34: Man nimmt ein **irreduzibles Polynom**[150] $f \in \mathbb{F}_p[x]$ und bildet den Körper $\mathbb{F}_p[x]/\langle f\rangle$, der dann ja $p^{\deg f}$ Elemente hat. Bei diesem Prozess werden die Nullstellen von f zu $\mathbb{F}_p$ **adjungiert**. Das macht man nun für alle irreduziblen Polynome über $\mathbb{F}_p$, die man ja nur modulo $x^p - 1$ betrachten muss, und erhält den algebraischen Abschluss

$$\overline{\mathbb{F}_p} := \bigcup_f \mathbb{F}_p/\langle f\rangle. \tag{6.6}$$

Zu einem vorgegebenen $n \in \mathbb{N}$ betrachten wir nun zwei Mengen über $\overline{\mathbb{F}_p}$ bzw. $E(\overline{\mathbb{F}_p})$, nämlich

$$E_n := \left\{\mathbf{x} \in E(\overline{\mathbb{F}_p}) \;:\; n\mathbf{x} = \mathbf{O}\right\} \qquad \text{und} \qquad \mu_n := \left\{x \in \overline{\mathbb{F}_p}^{\times} \;:\; x^n = 1\right\}. \tag{6.7}$$

[149] Aus kryptographischer Sicht.

[150] Das seine deg f Nullstellen ja eben nicht im Körper $\mathbb{F}_p$ hatte, denn sonst wäre es ja reduzibel, da man die zugehörigen linearen Polynome abdividieren könnte.

Das sind additive und multiplikative n**–te Einheitswurzeln**. Man kann zeigen, daß $E_n \simeq \mathbb{Z}_n \times \mathbb{Z}_n$, also insbesondere endlich ist, weswegen es ein $r > 0$ geben muß, so daß der relevante Teil $\overline{\mathbb{F}_p} \subseteq \mathbb{F}_{p^r}$ erfüllt und damit auch $E_n \subseteq E(\mathbb{F}_{p^r})$ ist. Das wesentliche Verbindungsglied ist die **Weil–Paarung**

$$e_n : E_n \times E_n \to \mu_n, \tag{6.8}$$

die sich benimmt wie eine Sesquilinearform, also ein Skalarprodukt über $\mathbb{C}$: Sie ist

1. **bilinear**
$$e_n(\mathbf{x} + \mathbf{x}', \mathbf{y}) = e_n(\mathbf{x}, \mathbf{y})\, e_n(\mathbf{x}', \mathbf{y}), \qquad e_n(\mathbf{x}, \mathbf{y} + \mathbf{y}') = e_n(\mathbf{x}, \mathbf{y})\, e_n(\mathbf{x}, \mathbf{y}').$$
2. **alternierend**
$$e_n(\mathbf{x}, \mathbf{y}) = e_n(\mathbf{y}, \mathbf{x})^{-1}.$$
3. **nichtentartet**
$$e_n(\mathbf{x}, \cdot) = 1 \qquad \Leftrightarrow \qquad \mathbf{x} = \mathbf{O}.$$
4. **Galois–äquivalent**
$$\mathbf{x}, \mathbf{y} \in E(\mathbb{F}_{p^r}) \qquad \Rightarrow \qquad e_n(\mathbf{x}, \mathbf{y}) \in \mathbb{F}_{p^r}^{\times}.$$

Wir brauchen uns gar nicht einbilden, auch nur die Existenz einer solchen Abbildung beweisen zu können, aber sowas gibt es und man kann es berechnen. Ausserdem ist für jedes feste $\mathbf{x}$ die Abbildung $e_n(\mathbf{x}, \cdot) : E_n \to \mu_n$ auch noch **surjektiv**, was in (Werner, 2002) sogar bewiesen wird.

Algorithmus 6.5 (MOV–Verfahren)
Gegeben: $\mathbf{x}, \mathbf{y} = k\mathbf{x} \in E(\mathbb{F}_p)$.

1. *Bestimme* $r > 0$ *mit* $E_n \subseteq E(\mathbb{F}_{p^r})$.
2. *Bestimme* $\mathbf{z} \in E(\mathbb{F}_{p^r})$, *so daß* $a := e_n(\mathbf{x}, \mathbf{z}) \in \mu_n$ *eine* ***primitive Einheitswurzel*** *ist, d.h.* $z^k \neq 1$ *für* $k < n$.
3. *Berechne* $y = e_n(\mathbf{y}, \mathbf{z})$.
4. *Bestimme* k *durch Lösung des Problems* $y = a^k$ *in* $\mathbb{F}_{p^r}^{\times}$, *beispielsweise mit der Indexkalkülmethode.*

Ergebnis: $k = \log_{\mathbf{x}} \mathbf{y}$.

Bemerkung 6.6

1. *Im MOV–Verfahren wird ausgenutzt, daß die Weil–Paarung das Problem von der komplexen elliptischen Kurve auf den wesentlich strukturierteren endlichen Körper* $\mathbb{F}_{p^r}$ *abbildet.*
2. *Man muss also bei der Wahl der elliptischen Kurve darauf achten, daß diese Abbildung nicht (relativ) einfach zu berechnen ist, genauso wie die Operationen in* $\mathbb{F}_{p^r}$.
3. *Das alles führt zu gewissen Bedinungen an das verwendete* p, *siehe (Werner, 2002).*

6 Literatur

Adleman, L. M., Pomerance, C., Rumely, R. S. (1983). On distinguishing prime numbers from composite numbers. *Ann. of Math.*, **117**:173–206.

Agrawal, M., Kayal, N., Saxena, N. (2004). PRIMES is in P. *Ann. of Math.*, **160**:781–793.

Alford, W. R., Grandville, A., Pomerance, C. (1994). There are infinitely many Charmichael numbers. *Ann. of Math.*, **140**:703–722.

Andrews, G. E. (1971). *Number Theory*. W. B. Saunders Company. Dover reprint 1994.

Diffie, W., Hellman, M. E. (1976). New directions in cryptography. *IEEE Trans. Inform. Theory*, **22**:644–654.

Gathen, J. v. z., Gerhard, J. (1999). *Modern Computer Algebra*. Cambridge University Press.

Hasse, H. (1933). Beweis des Analogons der Riemannschen vermutung für die Artinschen und F. K. Schmidtschen Kongruenzzetafunktionen in gewissen elliptischen Fällen. Vorläufige Mitteilung. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, **33**:253–262.

Karatsuba, A., Ofman, Y. (1963). Multiplication of multidigit numbers on automata. *Sovjet Physics–Doklady*, **7**:595–596.

Knuth, D. E. (1998). *The Art of Computer Programming. Seminumerical Algorithms*. Addison–Wesley, 3rd edition.

Köhler, W. (1987). *Die Blasinstrumente aus der Harmonie Universelle des Marin Mersenne*. Moeck.

McGeoch, C. (2001). Experimental analysis of algorithms. *Notices of the AMS*, **48**:304–311.

Mersenne, M. (2003). *Traité de l'harmonie universelle*. Corpus des œvres de philosophie in langue française. Arthème Fayard. Reprint, original Paris, 1627.

Poe, E. A. (1843). The gold bug. *Dollar Newspaper*.

Pollard, J. M. (1975). A Monte Carlo method for factorization. *BIT*, **15**:331–334.

Risen, A., Helmen, E. (1574). *Rechenbuch auff Linien und Ziphren*. Franck. Bey. Chr. Egen. Erben. Faksimile Th. Schäfer, Hannover, 1987.

Rivest, R. L., Shamir, A., Adleman, L. (1983). Cryptographic communications system and method. Patent US4405829, MIT.

Sauer, T. (2001). Computeralgebra. Vorlesungsskript, Justus–Liebig–Universität Gießen.

Sauer, T. (2013). Einführung in die Numerische Mathematik. Vorlesungsskript, Universität Passau. Online verfügbar, Lehrstuhlseite.

Schelter, W. (2001). Maxima - a Computer Algebra System. `http://maxima.sourceforge.net`.

Schönhage, A., Strassen, V. (1971). Schnelle Multiplikation großer Zahlen. *Computing*, **7**:281–292.

Schoof, R. (1985). Elliptic curves over fields and the computation of square roots mod p. *Math. Comp.*, **44**:483–494.

Sigler, L. (2002). *Fibonacci's Liber Abaci. Leoanardo Pisano's Book of Calculation*. Springer.

Singh, S. (2000a). *Fermats letzter Satz*. dtv. Deutsche Taschenbuchausgabe.

Singh, S. (2000b). *Geheime Botschaften*. dtv.

Stephenson, N. (1999). *Cryptonomicon*. Avon Books, New York.

Wätjen, D. (2008). *Kryptographie*. Spektrum Akademischer Verlag, 2. edition.

Werner, A. (2002). *Elliptische Kurven in der Kryptographie*. Springer.

Willems, W. (2008). *Codierungstheorie und Kryptograpie*. Birkhäuser.