Vorlesung Zahlentheorie II (Kryptographie) SS 2000

Inhaltsverzeichnis

Literatur

[1] Buchmann, J.: Einführung in die Kryptographie Springer, Berlin, 1999.

[2] Forster, O.: Algorithmische Zahlentheorie, Vieweg, Braunschweig, 1996.

[3] Koblitz, N.: A Course in number theory and cryptography Springer, New York, 1994.

[4] Kranakis, E.: Primality and cryptography, Wiley–Teubner, Stuttgart, 1986.

[5] Menezes, A. J., van Oorschot, P. C., Vanstone, S. A.: Handbook of applied cryptography, CRC Press, Stuttgart, 1997.

[6] Pomerance, C.: Cryptology and computational number theory, AMS, 1990.

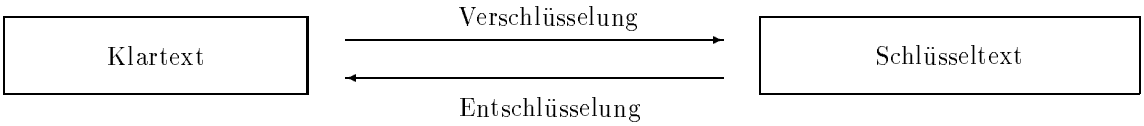
[7] Riesel, H.: Prime numbers and computer methods for factorization, Birkhäuser, Stuttgart, 1985.

[8] Stinson, D.: Cryptography, CRC Press, 1995.

Index

1 Einleitung

In der Kryptographie geht es um die Sicherheit von Daten. D. h. es werden Nachrichten verschlüsselt, um sie vor dem Zugriff Unbefugter zu schützen.



Historisch war dies vor allem für Militärs, Diplomaten und Kriminelle von Bedeutung. Doch im Zeitalter der elektronischen Kommunikation kommen viele weitere Anwendungen hinzu. Es sollen Methoden bereitgestellt werden, um folgende Ziele zu erreichen:

- **Privatheit, Geheimhaltung, Vertraulichkeit** Daten und Nachrichten (z. B. Telefongespräche, Faxe, Daten auf PC) sollen nur autorisierten Personen zugänglich sein.
- **Identifikation** Nachweis der Identität (z. B. bei einem Unix–Rechner oder einem Geldautomaten).
- **Authentifikation** Es soll gewährleistet sein, daß die Nachricht nicht von Fremden manipuliert wurde.
- **Integrität** von Daten und Software (z. B. Schutz vor Viren).
- **Signatur** Bei Vertragsschlüssen über elektronische Medien soll die Verbindlichkeit gewährleistet sein, d. h. im Zweifelsfall soll einem Dritten der Abschluß des Vertrages nachgewiesen werden können.
- **Anonymität** Beim Gebrauch von Bargeld kann (im Gegensatz zu Kreditkarten) das Konsumverhalten nicht nachverfolgt werden. Gesucht ist ein entsprechendes elektronisches Zahlungsmittel.

Spione oder Hacker sind daran interessiert, geheime Informationen zu erlangen bzw. zu manipulieren. Die Diskussion der Sicherheit von Verfahren gegenüber solchen Angriffen nennt man **Kryptoanalyse**. Dabei bestehen die folgende Möglichkeiten:

- **Ciphertext–Only Attacke** Der Angreifer kennt Schlüsseltexte und versucht daraus, die zugehörigen Klartexte bzw. den Schlüssel zu bestimmen.
- **Known–Plaintext Attacke** Der Angreifer kennt einen Klartext und den zugehörigen Schlüsseltext oder mehrere solche Paare. Er sucht den verwendeten Schlüssel.
- **Chosen–Plaintext Attacke** Der Angreifer kann die Schlüsseltexte zu selbst gewählten Klartexten erzeugen. Er kennt aber den Schlüssel nicht. Er sucht den verwendeten Schlüssel oder versucht aus dieser Kenntnis andere Schlüsseltexte zu entschlüsseln.

- **Adaptive Chosen–Plaintext Attacke** Der Angreifer kann die Schlüsseltexte zu selbst gewählten Klartexten erzeugen. Dabei kann er neue Klartexte in Abhängigkeit von den erhaltenen Schlüsseltexten wählen. Er kennt aber den Schlüssel nicht. Er sucht den verwendeten Schlüssel oder versucht aus dieser Kenntnis andere Schlüsseltexte zu entschlüsseln.

- **Chosen–Ciphertext Attacke** Der Angreifer kann selbst gewählte Schlüsseltexte entschlüsseln ohne den Schlüssel zu kennen. Er sucht den verwendeten Schlüssel.

Es sind eine Reihe von Methoden denkbar, diese Angriffe auszuführen. Eine einfache Ciphertext–Only Attacke besteht darin, alle möglichen Schlüssel auszuprobieren, bis der Klartext gefunden ist. Bei der Geschwindigkeit heutiger Computer führt diese Methode bei vielen einfachen Kryptosystemen zum Erfolg.
Im zweiten Weltkrieg wurden z. B. Hitlerreden, die öffentlich bekannt waren, bei Ihrer Übermittlung an U–Boote verschlüsselt. Dies ermöglichte den Alliierten einen Known–Plaintext–Angriff.
Eine wichtige Methode, Angriffe durchzuführen, besteht darin, statistische Eigenschaften einer Sprache auszunutzen (Häufigkeitsanalyse von einzelnen Buchstaben, Paaren oder Tripeln).

2 Komplexität von Verfahren

In der Kryptographie wird häufig mit sehr großen Zahlen gearbeitet. Der Aufwand von Verfahren (aber auch schon von Additionen und Multiplikationen) hängt natürlich von der Länge der zu verarbeitenden Zahlen ab. Diese misst man mit der Anzahl der benötigten Bits in der Binärentwicklung.

Definition 2.1 *Es sei $n \in \mathbb{N}$. Die* **Länge** *von n ist definiert durch $L(n) := [\log n] + 1$ (falls im* log *keine Basis angegeben ist, so ist immer $\log_2$ gemeint).*

Der Aufwand wird in der Regel als die benötigte Zeit (Anzahl der elementaren Operationen, **Bitoperationen**) betrachtet (der Speicheraufwand spielt heute meist nur noch eine untergeordnete Rolle). Das Zeitmaß hängt natürlich vom jeweiligen Computer ab, ist aber bei ein und demselben Computer proportional zu der Anzahl der ausgeführten elementaren Operationen bzw. Schritte. Technischer Fortschritt verbessert die Konstanten sehr schnell, so daß im wesentlichen nur die Größenordnung interessiert. Um diese zu erfassen, eignen sich die klassischen **Landau–Symbole**.

Definition 2.2 *Für $f, g : \mathbb{N} \to \mathbb{R}^+$ schreibt man*

(1) $f(n) = O(g(n))$, falls $c > 0$ und $n_0 \in \mathbb{N}$ existieren mit $f(n) \leq cg(n)$, für $n > n_0$.

(2) $f(n) = \Omega(g(n))$, falls $c > 0$ und $n_0 \in \mathbb{N}$ existieren mit $f(n) \geq cg(n)$, für $n > n_0$.

(3) $f(n) = \Theta(g(n))$, falls (1) und (2) gelten.

Eine andere Schreibweise für $f(n) = \Theta(g(n))$ ist $f(n) \asymp g(n)$. $\asymp$ stellt offensichtlich eine Äquivalenzrelation dar. Die Äquivalenzklasse von $f(n)$ heißt **Wachstumsrate** *von $f(n)$.*

Es gelten die folgenden Rechenregeln.

Satz 2.3 *(a) $f_1 = O(g)$ und $f_2 = O(g)$, dann gilt $f_1 + f_2 = O(g)$.*

(b) $f_1 = O(g_1)$ und $f_2 = O(g_2)$, dann gilt $f_1 \cdot f_2 = O(g_1 \cdot g_2)$.

(c) $g_1 \leq g_2$ und $f = O(g_1)$, dann gilt $f = O(g_2)$.

(d) $f_1 = O(f_2)$ und $f_2 = O(f_3)$, dann gilt $f_1 = O(f_3)$.

(e) $f_1 = O(g_1)$ und $f_2 = O(g_2)$, dann gilt $f_1 + f_2 = O(\max\{g_1, g_2\})$.

Beispiel: $L(n) = \Theta(\log n)$.

Die Dauer eines Verfahrens hängt einerseits von der Summe aller Längen der eingegebenen Daten, dem Input, ab. Andererseits ist die Zahl der ausgeführten Schritte auch abhängig vom Typ der eingegebenen Daten. Z. B. dauert es länger, 1011 mit 11111111 zu multiplizieren als mit 10000000. Darum geht man immer von dem ungünstigsten Fall (worst case) aus. Hat der Input die Länge n, so ist diese größtmögliche Anzahl eine Funktion von n, die die **Komplexität** des Verfahrens genannt wird (z. B. $2n^3$, $3n^2 + 4\log n$, $3^n + 6n^4$ etc.).
Als besonders effizient gelten die sogenannten polynomialen Algorithmen.

Definition 2.4 *Ein Algorithmus heißt* **polynomial**, *wenn es ein Polynom in der Länge des Input gibt, das den Aufwand des Algorithmus abschätzt. Die übrigen Algorithmen heißen* **exponentiell**.

Im folgenden wird der Aufwand für die Addition (Subtraktion), Mutliplikation und Division mit Rest angegeben. Dabei versuchen wir die Arbeitsweise des Computers (in naiver Weise) zu simulieren. Es seien a und b zwei natürliche Zahlen, gegeben durch ihre Binärentwicklungen mit Länge m bzw. n. Um $a+b$ zu berechnen werden die Zahlen untereinander geschrieben und bitweise mit Übertrag addiert. Dabei nimmt man an, daß die Addition zweier Bits (**Bitoperation**) eine Zeit $O(1)$ benötigt. Dann braucht die gesamte Addition Zeit $O(\max\{m,n\})$. Entsprechend kann man b von a in einer Zeit von $O(\max\{m,n\})$ subtrahieren.

Beispiel: $10101 + 111 = 11100$, $10101 - 111 = 1110$.

Bei der Berechnung von $a \cdot b$ geht man ähnlich vor. Man geht b von hinten nach vorn durch. Für jede 1 schreibt man a so auf, daß das letzte Bit von a unter der 1 von b steht. Dann addiert man dieses (verschobene) a zu dem bisherigen Ergebnis. Jede solche Addition kostet Zeit $O(m)$ und es gibt höchstens $O(n)$ Additionen. Damit benötigt die gesamte Multiplikation Zeit $O(mn)$.

Beispiel: $10101 \cdot 101 = 1101001$.

Bei der Division von a durch b mit Rest geht man wie in der Schule vor. Man schreibt b entsprechend verschoben unter a und subtrahiert. Mit dem Ergebnis verfährt man analog. Bei diesem Verfahren muß man $(m-n+1)$–mal Zahlen mit binärer Länge n voneinander abziehen. Dies kostet Zeit $O((m-n+1)n)$. Ist die binäre Länge des Quotienten k, so kann man wegen $m-n \le k \le m-n+1$ und $k \ge 1$ die Laufzeit durch $O(kn)$ abschätzen.

Beispiel: 10101 geteilt durch 101 ist 100 mit Rest 1.

Zusammenfassend halten wir fest

Satz 2.5 *Es seien $a, b \in \mathbb{Z}$.*

(a) Die Addition (und Subtraktion) von a und b erfordert Zeit $O(\max\{L(a), L(b)\})$.

(b) Die Multiplikation von a und b erfordert Zeit $O(L(a) \cdot L(b))$.

(c) Die Division von a durch b ($a = qb + r$) erfordert Zeit $O(L(b) \cdot L(q))$.

Beim Rechnen in dem Restklassenring $\mathbb{Z}/m$ werde eine Restklasse immer durch den kleinsten nichtnegativen Vertreter repräsentiert. Für den Aufwand der elementaren Operationen in $\mathbb{Z}/m$ gelten die folgenden Abschätzungen.

Satz 2.6 *Es sei $m \in \mathbb{Z}$.*

(a) Die Addition (und Subtraktion) von zwei Elementen aus $\mathbb{Z}/m$ erfordert Zeit $O(\log m)$.

(b) Die Multiplikation von zwei Elementen aus $\mathbb{Z}/m$ erfordert Zeit $O(\log^2 m)$.

(c) Die Bestimmung von $\operatorname{ggt}(a,b)$ *erfordert Zeit* $O(L(a)\cdot L(b))$.

(d) Das Feststellen, ob eine Restklasse prim ist und das Finden des Inversen erfordert Zeit $O(\log^2 m)$.

Oft wird die Potenz eines Elementes aus $\mathbb{Z}/m$ (oder allgemein a^n mit $n \in \mathbb{N}$ und $a \in G$ Element einer Gruppe) benötigt. Eine (aufwendige) Möglichkeit der Berechnung ist es, a mit sich selbst n mal zu multiplizieren. Eine effektivere Methode ist die sogenannte **schnelle Exponentation**. Dazu denkt man sich den Exponenten geschrieben als $n = \sum_{i=0}^{k} n_i 2^i$ (Binärentwicklung). Dann gilt

$$a^n = \prod_{i=0}^{k} (a^{2^i})^{n_i} = \prod_{0 \le i \le k,\ n_i = 1} a^{2^i}.$$

Durch fortgesetztes Quadrieren kann man also a^n berechnen.

Beispiel: 3^{73} in $G = (\mathbb{Z}/103\backslash\{0\}, \cdot)$: $3^{73} \equiv 37 \bmod 103$.

Algorithmus 2.7 (Power)
Eingabe: Gruppe G, $a \in G$, $n \in \mathbb{N}$.
Ausgabe: $a^n \in G$.

begin
  potenz := 1;
  while $n > 0$ **do**
  begin
    if n ungerade ($n_i = 1$) **then** potenz := potenz $\cdot a$;
    $n := n$ DIV 2 (letztes Bit abschneiden);
    $a := a \cdot a$;
  end;
  power := potenz;
end

Satz 2.8 *Algorithmus 2.7 berechnet* a^n *mit* $O(L(n))$ *Multiplikationen in* G. *Im Falle* $G = (\mathbb{Z}/m\backslash\{0\}, \cdot)$ *wird Zeit* $O(\log n \cdot \log^2 m)$ *benötigt.*

Sind $m_1, \ldots, m_k$ paarweise teilerfremd und $c_1, \ldots, c_k \in \mathbb{Z}$, so gibt der Chinesische Restsatz eine explizite Konstruktionsvorschrift für ein x mit $x \equiv c_i \bmod m_i$, für $i = 1, \ldots, k$.

Satz 2.9 *Der Chinesische Restsatz benötigt Zeit* $O(\log^2 m)$.

3 Endliche Körper

Viele kryptographische Verfahren beruhen auf Rechnungen in endlichen Körpern. Dazu werden algebraische Grundlagen gebraucht, die (zum Teil nur der Vollständigkeit halber) in diesem Kapitel aufgeführt werden sollen.

Definition 3.1 *Eine* **Gruppe** $(G, \cdot)$ *ist eine Menge* G *zusammen mit einer Verknüpfung* $\cdot : G \times G \to G$, *so daß*

(a) $(ab)c = a(bc)$, *für alle* $a, b, c \in G$,

(b) Es gibt ein neutrales Element $e \in G$, *so daß* $ae = ea = a$, *für alle* $a \in G$,

(c) Für alle $a \in G$ *gibt es ein inverses Element* $b = a^{-1} \in G$ *mit* $ab = ba = e$.

Gilt ferner $ab = ba$, *für alle* $a, b \in G$, *so heißt die Gruppe* **abelsch** *oder* **kommutativ**.

Ist $(G, \cdot)$ eine Gruppe und $H \subset G$, so daß $(H, \cdot)$ ebenfalls eine Gruppe ist, so wird diese **Untergruppe** genannt. Die Anzahl der Elemente einer endlichen Gruppe nennen wir die **Ordnung** der Gruppe. Ist $n \in \mathbb{N}$ die kleinste Zahl mit $a^n = e$, so heißt n die **Ordnung** des Elementes a in G (Bezeichnung: $\mathrm{ord}_G(a)$). In diesem Falle bildet $\langle a \rangle = \{e, a, a^2, \ldots a^{n-1}\}$ eine sogenannte **zyklische** Untergruppe mit **Erzeuger** a. Gilt $\langle a \rangle = G$, so heißt G **zyklische Gruppe** und a **Erzeuger** der Gruppe . Es gilt $a^k = e$ genau dann, wenn k ein Vielfaches von $\mathrm{ord}_G(a)$ ist. Ist G abelsch und H Untergruppe von G, so heißen die Mengen $aH := \{ah : h \in H\}$ die **Nebenklassen** von H. Da zwei Nebenklassen entweder identisch oder disjunkt sind, bilden sie eine Partition von G. Ferner haben alle Nebenklassen gleiche Anzahl von Elementen ($|H|$ Stück). Der Quotient $|G|/|H|$ wird **Index** von H in G genannt. Ein Element, das aus einer Nebenklasse gewählt wird, wird **Vertreter** derselben genannt. Man kann leicht nachprüfen, daß die Menge der Nebenklassen mit der Multiplikation $(aH)(bH) := (ab)H$ wieder eine Gruppe bildet, die sogenannte **Faktorgruppe**, die mit G/H bezeichnet wird.

Satz 3.2 *Es sei* G *eine endliche abelsche Gruppe mit Ordnung* $n \in \mathbb{N}$ *und* $a \in G$. *Dann gilt* $\mathrm{ord}_G(a) | n$.

Wieviele Erzeuger einer zyklischen Gruppe gibt es und wie kann man einen von ihnen bestimmen?

Satz 3.3 *Es sei* G *eine abelsche Gruppe mit Ordnung* $n \in \mathbb{N}$ *und* $a \in G$. *Ferner sei zu jedem Primteiler* p *von* n *die Zahl* $f(p) = f_a(p)$ *die größte mit* $a^{n/p^{f(p)}} = e$. *Dann gilt*

$$\mathrm{ord}_G(a) = \prod_{p|n} p^{\nu_p(n) - f(p)}.$$

Korollar 3.4 *Es sei* G *eine endliche abelsche Gruppe und* $a \in G$. *Gilt für* $k \in \mathbb{N}$ *sowohl* $a^k = e$ *als auch* $a^{k/p} \neq e$, *für alle Primteiler* p *von* k, *dann folgt* $k = \mathrm{ord}_G(a)$.

Dies liefert sofort einen Algorithmus zur Bestimmung eines Erzeugers einer zyklischen endlichen abelschen Gruppe.

Algorithmus 3.5 (Erzeuger)
Eingabe: Zyklische abelsche Gruppe der Ordnung n.
Ausgabe: Ein Erzeuger a von G.

begin
  Zerlege n in Primfaktoren: $n = p_1^{\alpha_1} \cdot \ldots \cdot p_r^{\alpha_r}$;
  repeat
    Bestimme $a \in G$ zufällig;
    $i := 1$;
    while $(a^{n/p_i} \neq e)$ **and** $(i \leq r)$ **do** $i := i + 1$;
    if $i = r + 1$ **then** erz := TRUE **else** erz := FALSE;
  until erz;
  $a \in G$ ist Erzeuger;
end

Eine wichtige Rolle in der Zahlentheorie spielt die **Eulersche φ–Funktion** $\varphi : \mathbb{N} \to \mathbb{N}$ definiert durch

$$\varphi(n) = |\{d : 1 \leq d \leq n-1,\ (d,n) = 1\}|.$$

Ihre wichtigsten Eigenschaften sind (vgl. Zahlentheorie I)

Satz 3.6 *(a)* $\varphi(n) = n \prod_{p|n}(1 - 1/p)$,

(b) $\sum_{d|n} \varphi(d) = n$,

(c) Falls $(a, m) = 1$, *dann gilt* $a^{\varphi(m)} \equiv 1 \bmod m$ *(Satz von Euler–Fermat).*

Satz 3.7 *Es sei G eine zyklische abelsche Gruppe mit Ordnung $n \in \mathbb{N}$. Dann gibt es zu jedem Teiler d von n genau $\varphi(d)$ Elemente der Ordnung d. Insbesondere gibt es $\varphi(n)$ Erzeuger.*

Man kann zeigen, daß $\varphi(n) \geq n/(6 \ln \ln n)$, für alle $n \geq 5$. Damit ist es sehr wahrscheinlich, daß Algorithmus 3.5 'schnell', d. h. mit wenigen Versuchen, einen Erzeuger findet. Ferner werden pro Test nur $O(r \log n) = O(\log^2 n)$ Multiplikationen benötigt.

Beispiel: Sei $G = P(101)$ die prime Restklassengruppe modulo 101 mit Ordnung $100 = 2^2 \cdot 5^2$. Die Restklasse $\overline{5}$ ist kein Erzeuger: $5^{100/2} = 5^{50} \equiv 1 \bmod 101$ und $5^{100/5} = 5^{20} \equiv 84 \bmod 101$. Die Restklasse $\overline{2}$ ist Erzeuger: $2^{50} \equiv -1 \bmod 101$ und $2^{20} \equiv -5 \bmod 101$.

Definition 3.8 *Eine Menge R zusammen mit zwei Verknüpfungen $\cdot$ und $+$ wird* **Ring** *genannt, falls*

(a) $(R,+)$ *ist abelsche Gruppe (mit neutralem Element 0),*

(b) $(ab)c = a(bc)$*, für alle* $a,b,c \in R$*,*

(c) $a(b+c) = ab+ac$ *und* $(a+b)c = ac+bc$*,*

Gilt $ab = ba$*, für alle* $a,b \in R$*, so heißt der Ring* **kommutativ**.

Zum Beispiel die ganzen Zahlen $\mathbb{Z}$ sind ein Ring.

Definition 3.9 *Es sei* $(R,+,\cdot)$ *ein Ring und* $\emptyset \neq S \subset R$*. Dann heißt* S *ein* **Ideal**, *falls*

(a) $a-b \in S$*, für alle* $a,b \in S$*,*

(b) $ab \in S$ *und* $ba \in S$*, für alle* $a \in S$ *und* $b \in R$*.*

Aus der zweiten Eigenschaft folgt insbesondere, daß $(S,+,\cdot)$ wieder ein Ring ist.

Definition 3.10 *Ein* **Körper** *ist ein Ring* $(R,+,\cdot)$*, für den* $(R\backslash\{0\},\cdot)$ *eine abelsche Gruppe ist.*

Jeder endliche, kommutative *nullteilerfreie* Ring ist auch ein Körper.

Satz 3.11 *Es sei* R *ein kommutativer endlicher Ring mit mindestens zwei Elementen. Gilt*

$$ab = 0 \Rightarrow (a = 0 \text{ oder } b = 0),$$

für alle $a,b \in R$ *(d. h.* R *ist nullteilerfrei), dann ist* R *ein Körper.*

Es sei S ein Ideal in dem Ring $(R,+,\cdot)$. Da $(S,+)$ eine Untergruppe der abelschen Gruppe $(R,+)$ ist, können wir die Faktorgruppe bilden. Hier werden die Nebenklassen **Restklassen** mod S genannt. Für diese Klassen kann eine Multiplikation in offensichtlicher Weise eingeführt werden

$$(a+S)(b+S) := ab+S.$$

Man beachte, daß diese Definition unabhängig von der Wahl der Vertreter a,b ist. Auf diese Weise erhält man einen Ring, den sogenannten **Restklassenring** R mod S, der mit R/S bezeichnet wird.
Das bekannteste Beispiel ist wohl $R = \mathbb{Z}$, $S = m\mathbb{Z}$ und $R/S = \mathbb{Z}/m$.
Aus der Zahlentheorie ist bekannt (und es kann mit Satz 3.11 leicht gezeigt werden), daß $\mathbb{Z}/m$ genau dann ein Körper ist, wenn m eine Primzahl ist.
Die ganzen Zahlen haben aufgrund der Existenz einer Division mit Rest die Eigenschaft, daß alle Ideale die Form $a\mathbb{Z} =: (a)$, mit $a \in \mathbb{Z}$ haben, d. h. es besteht aus den Vielfachen einer festen Zahl. Allgemein gilt

Definition 3.12 *Ein Ideal S eines Ringes R heißt* **Hauptideal**, *wenn es die Form $(a) = aR$ hat. Ein Ring, für den jedes Ideal ein Hauptideal ist, heißt* **Hauptidealring**.

Es sei nun $\mathbb{F}$ ein endlicher Körper. Die Menge $\mathbb{F}[x]$ der Polynome $a_0 + a_1x + \ldots + a_nx^n$ mit $n \in \mathbb{N}$ und Koeffizienten $a_i \in \mathbb{F}$ mit der üblichen Addition und Multiplikation bildet einen Ring. Wir nennen ein Polynom **normiert**, wenn es den Leitkoeffizienten 1 hat. Ein Polynom $g(x) \in \mathbb{F}[x]$ heißt **irreduzibel**, falls es keine Polynome $a(x), b(x) \in \mathbb{F}[x]$ mit kleinerem **Grad** gibt, so daß $g(x) = a(x)b(x)$. Man kann hier Polynomdivision wie gewohnt durchführen. Eine Folgerung ist, daß ein Polynom vom Grad n nicht mehr als n Nullstellen haben kann. Weitere Folgerungen sind (vgl. Zahlentheorie I)

Satz 3.13 *(a) $\mathbb{F}[x]$ ist ein Hauptidealring.*

(b) Sei $g(x) \in \mathbb{F}[x]$ irreduzibel. Dann ist $g(x)$ ein **Primelement**, *d. h. für alle $a(x), b(x) \in \mathbb{F}[x]$ mit $g(x)|(a(x)b(x))$ gilt $g(x)|a(x)$ oder $g(x)|b(x)$.*

(c) Jedes Polynom $f(x) \in \mathbb{F}[x]$ läßt sich (bis auf Konstanten) eindeutig in irreduzible Faktoren zerlegen.

Einen endlichen Körper mit q Elementen bezeichnen wir mit $\mathbb{F}_q$ oder $GF(q)$ (Galois Field). Wir kennen bereits einen Körper $\mathbb{F}_p$ für Primzahlen p. Weitere endliche Körper liefert

Satz 3.14 *Es sei $\mathbb{F}_q$ endlicher Körper mit q Elementen und $g(x)$ ein irreduzibles Polynom vom Grad r im Ring $\mathbb{F}_q[x]$. Dann ist der Restklassenring $\mathbb{F}_q[x]/(g(x))$ ein Körper mit q^r Elementen.*

Man kann zeigen, daß es irreduzible Polynome aus $\mathbb{F}_p[x]$ mit beliebigem Grad r gibt, so daß zu jeder Primzahlpotenz p^r ein Körper mit p^r Elementen existiert. Ferner ist bekannt, daß diese bis auf Isomorphie eindeutig sind und daß es keine weiteren endlichen Körper gibt (d. h. zu Nicht–Primzahlpotenzen). Zu $q = p^r$ meinen wir mit $\mathbb{F}_q$ immer diesen eindeutig bestimmten Körper und p heißt dessen **Charakteristik**.

Satz 3.15 *In $\mathbb{F}_q$ ist die multiplikative Gruppe $(\mathbb{F}_q\backslash\{0\}, \cdot)$ eine zyklische Gruppe.*

Ein erzeugendes Element dieser Gruppe (mit Ordnung $q-1$) heißt ein **primitives Element**. Im Falle $q = p$ und $\mathbb{F}_q = \mathbb{Z}/p$ sind die primitiven Elemente identisch mit den **Primitivwurzeln** modulo p (falls die prime Restklassengruppe $P(m)$ zyklisch ist, wird ein Erzeuger Primitivwurzel modulo m genannt). In diesem Fall zeigt Satz 3.15 also die Existenz von Primitivwurzeln modulo p. Darüberhinaus gilt

Satz 3.16 *Sei $m \in \mathbb{Z}$ mit $m = p^\alpha$ mit einer Primzahl $p > 2$ und $\alpha \geq 1$. Dann gibt es eine Primitivwurzel modulo m.*

Gauß zeigte, daß zu $m \in \mathbb{N}\backslash\{1\}$ genau dann Primitivwurzeln existieren, wenn $m = 2, 4,\ p^\alpha$ oder $2p^\alpha$ mit $\alpha \in \mathbb{N}$ und $p \in \mathcal{P}\backslash\{2\}$.

Lemma 3.17 *Es sei $\mathbb{F}_q$ ein Körper der Charakteristik p und $a, b \in \mathbb{F}_q$. Dann gilt $(a+b)^p = a^p + b^p$.*

Wie rechnet man in einem endlichen Körper $\mathbb{F}_q$ mit $q = p^r$? Es sei $g(x) \in \mathbb{F}_p[x]$ ein irreduzibles Polynom vom Grad r. Elemente von $\mathbb{F}_q$ werden gemäß Satz 3.14 als Polynome vom Grad $\leq r-1$ (mit r Koeffizienten aus $\mathbb{F}_p$) aufgefaßt. Man addiert bzw. multipliziert die Polynome und reduziert sie dann mittels Polynomdivision modulo $g(x)$. Die Koeffizienten kann man auch als Vektor $\in (\mathbb{F}_p)^r$ schreiben und mit komponentenweiser Addition und Skalarmultiplikation bilden sie einen $\mathbb{F}_p$–Vektorraum.

Beispiel: Das Polynom $x^4 + x + 1$ ist irreduzibel über $\mathbb{F}_2$. Der Körper $\mathbb{F}_{2^4}$ wird durch die Polynome mit Grad < 4 repräsentiert (vgl. Satz 3.14). Das Polynom x, das wir hier mit α bezeichnen wollen, ist ein primitives Element (siehe Tabelle). Man beachte, daß $\alpha^4 + \alpha + 1 = 0$. Jedes Element von$\mathbb{F}_{2^4}$ ist eine Linearkombination der Elemente $1, \alpha, \alpha^2$ und α^3. Wir erhalten die folgende Tabelle für $\mathbb{F}_{2^4}$.

0	$=$								$=$	$(0,0,0,0)$
1	$=$	1							$=$	$(1,0,0,0)$
α	$=$			α					$=$	$(0,1,0,0)$
α^2	$=$					α^2			$=$	$(0,0,1,0)$
α^3	$=$							α^3	$=$	$(0,0,0,1)$
α^4	$=$	1	$+$	α					$=$	$(1,1,0,0)$
α^5	$=$			α	$+$	α^2			$=$	$(0,1,1,0)$
α^6	$=$					α^2	$+$	α^3	$=$	$(0,0,1,1)$
α^7	$=$	1	$+$	α			$+$	α^3	$=$	$(1,1,0,1)$
α^8	$=$	1			$+$	α^2			$=$	$(1,0,1,0)$
α^9	$=$			α			$+$	α^3	$=$	$(0,1,0,1)$
α^{10}	$=$	1	$+$	α	$+$	α^2			$=$	$(1,1,1,0)$
α^{11}	$=$			α	$+$	α^2	$+$	α^3	$=$	$(0,1,1,1)$
α^{12}	$=$	1	$+$	α	$+$	α^2	$+$	α^3	$=$	$(1,1,1,1)$
α^{13}	$=$	1			$+$	α^2	$+$	α^3	$=$	$(1,0,1,1)$
α^{14}	$=$	1					$+$	α^3	$=$	$(1,0,0,1)$

Die Darstellung auf der rechten Seite zeigt, daß $\mathbb{F}_{2^4}$ als Vektorraum $(\mathbb{F}_2)^4$ aufgefaßt werden kann, wobei $\{1, \alpha, \alpha^2, \alpha^3\}$ eine Basis ist. Die linke Spalte eignet sich am besten zur Multiplikation (Exponenten addieren mod 15) und die rechte Spalte eignet sich am besten für die Addition (Vektoren addieren mod 2).
Im allgemeinen ist es schwierig, die eine Darstellung in die andere zu überführen. In der Praxis wird $\mathbb{F}_q$ durch die Vektorschreibweise dargestellt. Beispiel für eine Multiplikation: $a = (0,0,1,1)$, $b = (1,0,1,0)$, also $a(x) = x^3 + x^2$ und $b(x) = x^2 + 1$. Es ist $a \cdot b(x) = x^5 + x^4 + x^3 + x^2$. Polynomdivision durch $g(x) = x^4 + x + 1$ ergibt $x + 1$ mit Rest $x^3 + 1$. Damit ist $a \cdot b = (1,0,0,1)$.

Satz 3.18 *Sei $\mathbb{F}_q$ endlicher Körper mit $q = p^r$. Dann gilt*

(a) Zwei Elemente können in einer Zeit von $O(\log q)$ addiert werden.

(b) Zwei Elemente können in einer Zeit von $O(\log^2 q)$ multipliziert werden.

(c) Die Inversion eines Elementes oder eine Division benötigen eine Zeit von $O(\log^3 q)$.

(d) Für $k \in \mathbb{N}$ kann die k–te Potenz von einem $a \in \mathbb{F}_q$ in einer Zeit von $O(\log k \log^2 q)$ bestimmt werden.

4 Einige klassische Kryptosysteme

4.1 Grundlegende Begriffe

Zunächst geben wir einige grundlegende Begriffe der Kryptographie an.

Definition 4.1 *(a) Eine endliche, nichtleere Menge $\mathcal{A} = \{a_0, \ldots, a_{n-1}\}$ heißt* **Alphabet** *(z. B. das lateinische Alphabet mit Leerzeichen und ohne Sonderzeichen $\mathcal{L} = \{\square, A, B, \ldots, X, Y, Z\}$ oder $\mathcal{A} = \{0, 1\}$ oder der Ascii–Zeichensatz).*

(b) Ihre Elemente heißen **Zeichen** *(***Buchstaben, Symbole***).*

(c) Die Anzahl der Elemente eines Alphabets heißt **Länge** *des Alphabets (Schreibweise: $|\mathcal{A}| = n$).*

(d) Ein **Wort** *ist eine endliche Folge von Zeichen. Die Menge aller Worte über einem Alphabet $\mathcal{A}$ wird mit $\mathcal{A}^*$ bezeichnet.*

Meist enthält das Alphabet ein Leerzeichen, mit dem Wörter getrennt werden. Es folgt die allgemeine und abstrakte Definition eines Kryptosystems.

Definition 4.2 *Ein* **Verschlüsselungsverfahren** *(***Kryptosystem, Chiffre***) ist ein Fünftupel $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit folgenden Eigenschaften:*

(a) $\mathcal{P}$ ist eine Menge, ihre Elemente heißen Klartexte, $\mathcal{P}$ heißt **Klartextraum**.

(b) $\mathcal{C}$ ist eine Menge, ihre Elemente heißen Schlüsseltexte, $\mathcal{C}$ heißt **Schlüsseltextraum**.

(c) $\mathcal{K}$ ist eine Menge, ihre Elemente heißen Schlüssel, $\mathcal{K}$ heißt **Schlüsselraum**.

*(d) $\mathcal{E} = \{E_e : e \in \mathcal{K}\}$ ist eine Familie von (injektiven) Funktionen $E_e : \mathcal{P} \to \mathcal{C}$ (***Verschlüsselungsfunktionen, Codes***).*

*(e) $\mathcal{D} = \{D_d : d \in \mathcal{K}\}$ ist eine Familie von Funktionen $D_d : \mathcal{C} \to \mathcal{P}$ (***Entschlüsselungsfunktionen***).*

(f) Zu jedem $e \in \mathcal{K}$ gibt es ein $d \in \mathcal{K}$ mit $D_d(E_e(x)) = x$, für alle $x \in \mathcal{P}$.

Zur besseren Verarbeitung wird der Text in Blöcke einer gewissen Länge l von Symbolen unterteilt (falls l die Textlänge nicht teilt, füllt man mit Leerzeichen auf). Ein Text läßt sich also als eine endliche Folge von Blöcken aus $\mathcal{A}^l$ auffassen. Die Verschlüsselungsfunktion (der Code) ist dann gegeben durch eine injektive Abbildung von $\mathcal{P} = \mathcal{A}^k$ nach $\mathcal{C} = \mathcal{A}^l$.
Um mit den Ver– und Entschlüsselungsfunktionen praktisch rechnen zu können, werden die Buchstaben zunächst in Zahlen (oder auch in Elemente eines endlichen Körpers) verwandelt. Dies geschieht mit einer bijektiven Abbildung

$$\beta : \mathcal{A}^l \to \{0, 1, \ldots, N-1\}^l \sim (\mathbb{Z}/N)^l,$$

wobei $N = |\mathcal{A}|$ (wir setzen für unsere Zwecke die Menge $\{0, 1, \ldots, N-1\}$ gleich mit dem Restklassenring $\mathbb{Z}/N$).

Beispiel: $\lambda : \mathcal{L} \to \mathbb{Z}/27$ mit $\lambda(\square) = 0,\ \lambda(\text{ A }) = 1, \ldots,\ \lambda(\text{ Z }) = 26$
oder $\lambda^l : \mathcal{L}^l \to (\mathbb{Z}/27)^l$ mit $\lambda^l(a_1, \ldots, a_l) = (\lambda(a_1), \ldots, \lambda(a_l))$.
Zum Beispiel geht das Wort BOND über in 2 15 14 4.

Um Vektoren in Zahlen zu verwandeln und umgekehrt, eignet sich der **m–adische Code:**

$$\vartheta_{m,l} : (\mathbb{Z}/m)^l \to \mathbb{Z}/m^l, \quad \vartheta_{m,l}(a_1, \ldots, a_l) := \sum_{i=1}^{l} a_i m^{i-1}.$$

Beispiel: $m = 2$: $(1, 0, 1, 1)$ geht über in 1101 (=13).
$m = 27$: $(2, 15, 14, 4)$ geht über in $2 + 15 \cdot 27 + 14 \cdot 27^2 + 14 \cdot 27^3 + 4 \cdot 27^4 = 2411939$.

Mittels des **Inklusionscodes**

$$i_{m,n} : \mathbb{Z}/m \to \mathbb{Z}/n, \quad i_{m,n}(x) = x$$

kann man die Zahlen von 0 bis m auch als Elemente des Restklassenrings $\mathbb{Z}/n$, mit $n > m$ auffassen.

Es sei eine Ver– oder Entschlüsselungsfunktion f vom Typ $\mathbb{Z}/m \to \mathbb{Z}/m$ gegeben. Ferner sei $\mathcal{A}$ ein Alphabet der Länge N. und l und L seien die Längen der Klartext– bzw. Schlüsseltextblöcke mit $N^l \leq m \leq N^L$. Daraus kann man ein f^* vom Typ $\mathcal{A}^l \to \mathcal{A}^L$ wie folgt konstruieren:

$$f^* = (\beta^L)^{-1} \circ (\vartheta_{N,L})^{-1} \circ i_{m,N^L} \circ f \circ i_{N^l,m} \circ \vartheta_{N,l} \circ \beta^l$$

4.2 Die Verschiebungschiffre

In diesem einfachen Beispiel ist $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}/m$. Für einen Schlüssel $K \in \mathcal{K}$ sind e_K und d_K definiert durch

$$E_K(x) = x + K \bmod m \quad \text{und} \quad D_K(y) = y - K \bmod m.$$

Dieses Verschlüsselungsverfahren soll schon von Cäsar benutzt worden sein.

Beispiel: $m = 27,\ K = 7$

V	E	N	I		V	I	D	I		V	I	C	I
22	5	14	9	0	22	9	4	9	0	22	9	3	9
2	12	21	16	7	2	16	11	16	7	2	16	10	16
B	L	U	P	G	B	P	K	P	G	B	P	J	P

Die Anzahl der möglichen Schlüssel ist m. Diese Art von Verschlüsselung läßt sich aber schon mit etwas Sprachgefühl und Probieren knacken.

4.3 Die Substitutionschiffre

Hier wird jedes Symbol des Alphabets durch ein anderes ersetzt. Es ist $\mathcal{P} = \mathcal{C} = \mathbb{Z}/m$ und $\mathcal{K}$ ist die Menge der bijektiven Abbildungen π von $\mathbb{Z}/m$ nach $\mathbb{Z}/m$. Für $\pi \in \mathcal{K}$ ist

$$E_\pi(x) = \pi(x) \quad \text{und} \quad D_\pi(y) = \pi^{-1}(y).$$

Das Entschlüsseln entspricht hier dem Nachschlagen in einer Tabelle und dem Ersetzen durch einen anderen Buchstaben.

Beispiel: $m = 27$ und π gegeben durch

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

Hierbei geht z. B. TASTATUR über in YQLYQYXK.

Die Anzahl der möglichen Schlüssel ist $m!$. Sukzessives Probieren der Schlüssel ist bei großer Tabelle also unmöglich. Dennoch ist die Substitutionschiffre nicht sicher (siehe Abschnitt 4.7).

4.4 Die affine Chiffre

Hier ist $\mathcal{P} = \mathcal{C} = \mathbb{Z}/m$ und $\mathcal{K} = \{(a, b) \in (\mathbb{Z}/m)^2 : \text{ ggt}\,(a, m) = 1\}$. Für ein $K = (a, b) \in \mathcal{K}$ ist

$$E_{(a,b)}(x) = a \cdot x + b \bmod m \quad \text{und} \quad D_{(a,b)}(y) = a^{-1}(y - b) \bmod m.$$

Dabei existiert das Inverse a^{-1} modulo m genau dann, wenn $\text{ggt}\,(a, m) = 1$.

Beispiel: $m = 27$, $K = (7, 3)$. Das Inverse von 7 modulo 27 ist 4 (da $4 \cdot 7 \equiv 1 \bmod 27$).

A	F	F	I	N
1	6	6	9	14
10	18	18	12	20
J	R	R	L	T

Im Fall $a = 1$ erhält man die Verschiebungscodes und im Fall $b = 0$ spricht man von **Linearen Codes**. Die Anzahl der möglichen Schlüssel ist $\varphi(m) \cdot m$. Bei längeren Botschaften kann man leicht den Buchstaben bestimmen, der am häufigsten vorkommt. Bei deutschen Texten ist dies dann die Verschlüsselung von 'E'. Verfährt man analog mit dem zweithäufigsten Buchstaben, so kann man aus diesen Informationen schon den Schlüssel (a, b) durch Lösen eines Linearen Kongruenzensystems bestimmen.

4.5 Die Vigenère–Chiffre

Hier ist $\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}/m)^n$ und für $K = (k_1, \ldots, k_n) \in \mathcal{K}$ ist

$$E_K(x_1, \ldots, x_n) = (x_1 + k_1, \ldots, x_n + k_n) \bmod m$$

und

$$E_K(y_1, \ldots, y_n) = (y_1 - k_1, \ldots, y_n - k_n) \bmod m.$$

Der Schlüssel ist hier durch ein **Codewort** K gegeben.

Beispiel: $m = 27,\ n = 3,\ K = (\ \mathrm{M}\quad \mathrm{O}\quad \mathrm{T}\ ) = (13, 15, 20)$.

C	O	G	I	T	O		E	R	G	O		S	U	M
3	15	7	9	20	15	0	5	18	7	15	0	19	21	13
16	3	0	22	8	8	13	20	11	20	3	20	5	9	6
P	C		V	H	H	M	T	K	T	C	T	E	I	F

Die Anzahl der möglichen Schlüssel ist m^n. Bei bekannter Blocklänge kann man anhand der n Teiltexte getrennt $k_1, \ldots, k_n$ analog wie bei der affinen Chiffre bestimmen.

4.6 Die Hill–Chiffre

Hier ist $\mathcal{P} = \mathcal{C} = (\mathbb{Z}/m)^n$ und

$$\mathcal{K} = \{A \in \mathbb{Z}^{n \times n} : (\det A, m) = 1\}.$$

Für $A \in \mathcal{K}$ und $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n)$ ist

$$E_A(x) = Ax \quad \text{und} \quad D_A(y) = A^{-1}y.$$

Hierbei ist A^{-1} die modulo m eindeutig bestimmte Matrix mit $A \cdot A^{-1} \equiv I \bmod m$. Die Existenz der inversen Matrix wird gewährleistet durch

Satz 4.3 *Sei $A = (a_{ij})_{1 \le i,j \le n}$ eine Matrix mit $a_{ij} \in \mathbb{Z}/m$ und $D = \det A$. Dann sind die folgenden Aussagen äquivalent:*

(a) $\mathrm{ggt}\,(D, m) = 1$.

(b) A ist invertierbar.

(c) $x \mapsto A \cdot x$ ist eine bijektive Abbildung $(\mathbb{Z}/m)^n \to (\mathbb{Z}/m)^n$.

(d) Für $x \in (\mathbb{Z}/m)^n \backslash \{0\}$ gilt $A \cdot x \neq 0$.

(e) Die Spalten (Zeilen) von A sind modulo m linear unabhängig.

Bemerkung: Im Falle $(\det A, m) = 1$ ist die Inverse von A modulo m

$$A^{-1}_{\mathrm{mod}m} = (\det A)^{-1}_{\mathrm{mod}m} \cdot \det A \cdot A^{-1}_{\mathbb{R}}.$$

Man beachte, daß $\det A \cdot A^{-1} = \mathrm{adj(A)}$, die **Adjunkte** ganzzahlig ist.

Beispiel: $m = 27,\ n = 2,\ A = \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix},\ D \equiv -1 \bmod 27,\ A^{-1} = \begin{pmatrix} 20 & 8 \\ 3 & 16 \end{pmatrix}.$

M	A	T	R	I	X	C	O	D	E
13	1	20	18	9	24	3	15	4	5
16	19	13	24	21	6	18	6	3	20
P	S	M	X	U	F	R	F	C	T

Satz 4.4 *Die Anzahl der möglichen Schlüssel für die Hill–Chiffre ist*

$$m^{n^2} \prod_{p|m} \left(1 - \frac{1}{p}\right) \cdot \left(1 - \frac{1}{p^2}\right) \cdot \ldots \cdot \left(1 - \frac{1}{p^n}\right).$$

Die Hill–Chiffre kann leicht durch einen Known–Plaintext Angriff gebrochen werden. Angenommen, man kennt n Klartexte $x_1, \ldots, x_n$ und die zugehörigen Schlüsseltexte $y_i = A \cdot x_i$. Ist $X = (x_1, \ldots, x_n)$ und $Y = (y_1, \ldots, y_n)$, dann gilt $A \cdot X \equiv Y \bmod m$. Gilt jetzt noch $\mathrm{ggt}\,(\det X, m) = 1$, so bestimmt man die Inverse von X modulo m und erhält $A \equiv Y \cdot X^{-1} \bmod m$. Im Falle $\mathrm{ggt}\,(\det Y, m) = 1$, erhält man die Entschlüsselungsmatrix durch $A^{-1} \equiv X \cdot Y^{-1} \bmod m$.

4.7 Die Permutationschiffre

Hier gilt $\mathcal{P} = \mathcal{C} = (\mathbb{Z}/m)^n$. $\mathcal{K}$ ist die Menge Σ_n der Permutationen von n Elementen. Für $\pi \in \mathcal{K}$ ist

$$E_\pi(x_1, \ldots, x_n) = (x_{\pi(1)}, \ldots, x_{\pi(n)})$$

und

$$D_\pi(y_1, \ldots, y_n) = (y_{\pi^{-1}(1)}, \ldots, y_{\pi^{-1}(n)}).$$

Es werden also jeweils Blöcke der Länge n permutiert.

Beispiel: $m = 27,\ n = 5,\ \pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 1 & 5 & 3 & 2 \end{pmatrix}.$

H	I	E	R		I	S	T		A	L	L	E	S		V	E	R	T	A	U	S	C	H	T
R	H		E	I		I	A	T	S	S	L		E	L	T	V	A	R	E	H	U	T	C	S

Die Anzahl der möglichen Schlüssel ist $n!$. Die Permutationschiffre kann ebenfalls leicht gebrochen werden:

Satz 4.5 *Die Permutationschiffre ist ein Spezialfall der Hill–Chiffre.*

5 Public–Key–Verfahren

5.1 Die Idee

Die bisher behandelten Kryptosysteme sind **symmetrisch**, d. h. aus der Kenntnis der Verschlüsselungsfunktion e kann die Entschlüsselungsfunktion d ohne großen Aufwand bestimmt werden und umgekehrt. Dies führt zu Problemen bei der Verteilung und Verwaltung von Schlüsseln. Immer, wenn zwei Personen miteinander geheim kommunizieren wollen, müssen sie vorher einen geheimen Schlüssel austauschen. Dafür muß aber ein sicherer Kanal zur Verfügung stehen. Bei einem Benutzerkreis von n Personen müssten dann $n(n-1)/2$ geheime Schlüssel übertragen werden. Dies ist aber für große n organisatorisch kaum zu bewältigen.
In einem Public–Key System braucht man statt eines Schlüssels ein Schlüsselpaar (e, d) (e zum Verschlüsseln, d zum Entschlüsseln). Dabei kann d ohne zusätzliche Information nicht 'in vertretbarer Zeit' aus e berechnet werden. Dazu bedient man sich sogenannter **Falltürfunktionen**. Eine solche Funktion kann relativ schnell berechnet werden, die Umkehrfunktion dagegen ohne zusätzliche Information nur in 'unakzeptabler Zeit'. Der Begriff 'unakzeptabel' ist dabei abhängig vom Stand der Technik und der bekannten Verfahren. Es empfiehlt sich darum, das Verfahren an ein bekanntes mathematisches Problem zu koppeln, bei dem ein wesentlicher Fortschritt schnell öffentlich wird.

Beispiel: (1) Die Multiplikation zweier Primzahlen p und q.
(2) Die Exponentation in einer endlichen Gruppe G.

Beispiel (1) führt zu dem Problem der Faktorisierung von ganzen Zahlen, während Beispiel (2) zu dem Problem des **Diskreten Logarithmus** führt:

Definition 5.1 *Sei G eine endliche Gruppe und $g \in G$. Ein $a \in \mathbb{N}$ heißt* **Diskreter Logarithmus** *von $A \in G$, wenn $g^a = A$.*

Das Prinzip bei Public–Key Verfahren ist dann wie folgt. Jeder Teilnehmer kann seinen Verschlüsselungsschlüssel veröffentlichen (e heißt öffentlicher Schlüssel, Public–Key). Den Entschlüsselungsschlüssel d hält er geheim (d heißt privater Schlüssel, Private Key). Die öffentlichen Schlüssel werden in einer frei zugänglichen Datenbank (vgl. Telefonbuch) abgelegt. Will Teilnehmer i an Teilnehmer j eine geheime Botschaft schicken, so verwendet er den öffentlichen Schlüssel e_j. Die Nachricht kann nur mit Kenntnis von d_j entschlüsselt werden, also nur von Teilnehmer j.

Elektronische Signaturen: Es muß gewährleistet sein, daß die öffentlichen Schlüssel nicht gefälscht werden können. Dies geschieht mit elektronischen Signaturen, die z. B. mit Public–Key Verfahren erstellt werden können. Es seien n Teilnehmer mit Schlüsselpaaren $(e_1, d_1), \ldots, (e_n, d_n)$ gegeben. Der Einfachheit halber nehmen wir an, daß für alle Teilnehmer der Klartextraum $\mathcal{P}_i$ und der Schlüsseltextraum $\mathcal{C}_i$ gleich und für alle Teilnehmer die selben sind. Will Teilnehmer i an j eine geheime Nachricht schicken und j soll sicher sein, daß sie von i stammt, so verschlüsselt i seine Nachricht mit e_j. Am Ende überträgt er seine Unterschrift $U \in \mathcal{P}$ (evtl. mit einer Identifikationsnummer, Sendezeit der Nachricht etc.). Diese verschlüsselt er zunächst mit d_i und dann mit e_j. Teilnehmer j erhält also die verschlüsselte

Nachricht zusammen mit $e_j(d_i(U))$. Zum Entschlüsseln benutzt er d_j und erhält die Nachricht mit einem unverständlichen Block am Ende. Diesen Block kann er dann mit e_i entschlüsseln und erhält U. Teilnehmer j ist dann sicher, daß der Sender Teilnehmer i war.

Hashfunktionen: Zur Garantie, daß die übertragene Nachricht nicht von Unbefugten verändert wurde, bedient man sich sogenannter Hash–Funktionen. Grob gesprochen ist eine Hash–Funktion eine einfach zu berechnende Funktion $f : x \mapsto h$, die einem sehr großen Input x (z. B. Strings von 10^6 Bit) einen wesentlich kleineren Output h (String von 150 oder 200 Bit) zuordnet. Dabei soll f 'algorithmisch injektiv' sein, d. h. es ist mit akzeptablen Rechenaufwand nicht möglich, ein x' zu finden mit $f(x') = f(x)$. Ist N die Nachricht, so wird $h = f(N)$ der Unterschrift U hinzugefügt. Der Empfänger berechnet $f(N)$ und vergleicht ihn mit dem Wert aus U. Bei Übereinstimmung weiß er, daß die Nachricht nicht manipuliert wurde.

Effizienz: Leider sind die bekannten Public–Key Verfahren nicht so effizient wie viele symmetrische Verfahren. Darum benutzt man in der Praxis eine Kombination aus beiden. Das Public–Key Verfahren wird z. B. verwendet, um die (relativ kurzen) Schlüssel auszutauschen. Die großen Datenmengen werden dann mit einem symmetrischen Verfahren verschlüsselt.

5.2 RSA

Im folgenden stellen wir als ein Beispiel für ein Public–Key System das RSA–System (Rivest, Shamir, Adleman, 1977) vom Typ $\mathbb{Z}/m \to \mathbb{Z}/m$ vor.

(1) Wähle zwei verschiedene *große* 'Zufallsprimzahlen' p und q.
Setze $m := p \cdot q$.

(2) Wähle eine 'Zufallszahl' e, $1 < e < \varphi(m) = (p-1)(q-1)$, mit $(e, \varphi(m)) = 1$.

(3) Sei $C_{m,e} : \mathbb{Z}/m \to \mathbb{Z}/m$ definiert durch $C_{m,e}(x) = x^e$.

(4) Berechne $(\bar{e})^{-1}$ in $\mathbb{Z}/\varphi(m)$; $(\bar{e})^{-1} = \bar{d}$, mit $1 < d < \varphi(m)$, d. h. $d \cdot e \equiv 1 \bmod \varphi(m)$.
Ferner sei $C_{m,d} : \mathbb{Z}/m \to \mathbb{Z}/m$ definiert durch $C_{m,d}(y) = y^d$.

$C_{m,e}$ und $C_{m,d}$ können 'schnell' mit Algorithmus 2.7 berechnet werden. m und e werden veröffentlicht. Dagegen bleiben p, q und d geheim.

Lemma 5.2 *Es gilt* $C_{m,d} \circ C_{m,e} = \mathrm{id}\,(\mathbb{Z}/m)$, *d. h.* $C_{m,e}$ *ist bijektiv mit* $C_{m,e}^{-1} = C_{m,d}$.

Formal ist also das RSA–Kryptosystem gegeben durch $\mathcal{P} = \mathcal{C} = \mathbb{Z}/m$ und

$$\mathcal{K} = \{(m, p, q, e, d) : m = pq,\ de \equiv 1 \bmod (p-1)(q-1)\}$$

und für $K = (m, e, p, q, e, d)$ durch

$$E_K(x) = x^e \quad \text{und} \quad D_K(y) = y^d.$$

Benutzung von RSA: Es sei $\mathcal{A}$ ein Alphabet und $N = |\mathcal{A}|$. Man gebe zunächst l und L so vor, daß $N^{l/2}$ und $N^{L/2}$ 'vernünftige' Schranken für p und q sind: $N^{l/2} \leq p, q \leq N^{L/2}$ Jeder Teilnehmer wählt Primzahlen p_i, q_i in diesem Bereich und erhält so ein $m = m_i$ mit $N^l \leq m \leq N^L$. Wie in Abschnitt 4.1 beschrieben, erhält man die Verschlüsselungsfunktion $C^*_{m,e} : \mathcal{A}^l \to \mathcal{A}^L$. Falls m und d bekannt sind, kennt man auch die Entschlüsselungsfunktion $(C^*_{m,e})^{-1}$.

Beispiel: Eine Botschaft (im lateinischen Alphabet $\mathcal{L}$) wurde mittels RSA Verfahren verschlüsselt. Dabei war $N = 27,\ l = 1,\ L = 2,\ m = 187,\ e = 107$ und

$$C^*_{187,107} = (\lambda^2)^{-1} \circ (\vartheta_{27,2})^{-1} \circ i_{187,27^2} \circ C_{187,107} \circ i_{27^1,187} \circ \vartheta_{27,1} \circ \lambda$$

Die verschlüsselte Botschaft lautet `_E__QCAFGD__DC__VAQC__CFW_EDO_AF.`
Man ermittelt leicht $p = 11,\ q = 17$ und $d = 3$. Durch Anwendung von $(C^*_{187,107})^{-1}$ findet man dann den Klartext `P_UND_Q_ZU_KLEIN.`

RSA und Faktorisierung: RSA beruht darauf, daß es bis heute sehr schwierig ist, die Zahl m zu faktorisieren, d. h. in Primfaktoren zu zerlegen. Die bis heute bekannten Verfahren benötigen bei hinreichend großen Primzahlen p und q einen unakzeptablen Rechenaufwand. Die Rechenzeit für die Faktorisierung einer Zahl mit 200 Dezimalstellen beträgt nach heutiger Kenntnis der schnellsten Algorithmen ca. 10^9 Jahre, ist also in der Praxis unmöglich.
Bis heute ist nicht bewiesen, daß die einzige Möglichkeit, RSA zu brechen, die Faktorisierung von m ist. Theoretisch könnte es einen Weg geben, den Klartext zu bestimmen, ohne d zu kennen.

Jedoch ist die Kenntnis von d äquivalent zu der von p und q. Denn einerseits kann d aus p und q bestimmt werden. Und andererseits kann man aus d mit wenig Aufwand p und q bestimmen. Hierfür wollen wir ein probabilistisches Verfahren angeben.

Sei

$$k = ed - 1,$$

also ein Vielfaches von $\varphi(m) = (p-1)(q-1)$, insbesondere also k gerade. Ferner sei

$$k = 2^s r, \quad \text{mit } r \text{ ungerade und } s \geq 2.$$

Findet man ein $a \in \{1, \ldots, m-1\}$ und ein $t \in \{1, \ldots, s\}$ mit

$$1 < g := \text{ggt}\,(a^{k/2^t} - 1, m) < m,$$

so ist g ein echter Teiler von m, also p oder q. Ein solches a wird **Teilererzeuger** genannt.

Beispiel: $m = 187,\ e = 107$ und $d = 3$. Dann ist $k = 320 = 64 \cdot 5 = 2^6 \cdot 5$, also $r = 5$ und $s = 6$.
Mit $a = 2$ ergibt sich für $t = 6$ also $a^5 - 1 = 31$ und $g = 1$. Für $t = 5$ erhält man $a^{10} - 1 = 1023 = 3 \cdot 11 \cdot 31$ und damit $g = 11$. (Für $t = 4$ ist $g = 11$ und für $t = 3, 2, 1$ ist $g = 187$.)

Satz 5.3 *In der Menge* $\{1, \ldots, m-1\}$ *gibt es mindestens* $\varphi(m)/2$ *Teilererzeuger.*

Man wählt nun zufällig und gleichverteilt eine Zahl $a \in \{0, \ldots, m-1\}$. Dann berechnet man $\operatorname{ggt}(a, m)$. Falls dieser > 1 ist, so hat man einen echten Teiler gefunden. Ansonsten prüft man, ob a ein Teilererzeuger ist. Die Wahrscheinlichkeit, dabei einen Primteiler von m zu finden ist nach Satz 5.3 mindestens $1/2$. Also findet man mit großer Wahrscheinlichkeit mit wenigen Wiederholungen einen Primteiler von m.

Weiter ist die Kenntnis von p, q (im wesentlichen) äquivalent zu der von $\varphi(m)$:

Satz 5.4 *Es sei m Produkt von zwei ungeraden Primzahlen p und q. Dann ist die Kenntnis der Zerlegung von m äquivalent zur Kenntnis von $\varphi(m)$. Genauer gilt: $\varphi(m)$ kann aus p und q in einer Zeit von $O(\log m)$ bestimmt werden und p und q können aus $\varphi(m)$ in einer Zeit von $O(\log^3 m)$ bestimmt werden.*

Auswahl der Parameter p, q, e: Die Primzahlen p und q sollten so gewählt sein, daß kein bekanntes Faktorisierungsverfahren schnell zum Erfolg kommt. Darum sollten p und q

- nicht zu klein sein (mindestens 512 Bits, besser mehr).
- ungefähr gleich sein.
- doch nicht zu gleich sein, da man sonst mit der Fermat–Faktorisierung schnell zum Ziel kommt.
- so sein, daß $(p-1)$ und $(q-1)$ große Primfaktoren enthalten (sonst greift das $(p-1)$–Verfahren von Pollard).
- keine bekannten Primzahlen, z. B. Mersennesche, sein.

Am wirkungsvollsten ist es, p und q zufällig zu wählen.

Der Exponent e sollte nicht zu klein sein, da man sonst leicht einen Angriff starten kann. Ist z. B. $e = 3$ (das ist kleinstmöglich), und wird eine Nachricht x zu drei teilerfremden Moduln m_1, m_2, m_3 verschlüsselt (z. B. ein Serienbrief und alle Empfänger haben $e = 3$), so kennt man die drei Schlüsseltexte $y_i \equiv x^3 \bmod m_i,\ i = 1, 2, 3$. Mit $y = x^3$ gilt

$$y \equiv y_i \bmod m_i,\ i = 1, 2, 3 \quad \text{und} \quad 0 \le y = x^3 < m_1 m_2 m_3.$$

Mit Hilfe des chinesischen Restsatzes kann man y bestimmen. x erhält man durch Ziehen der Wurzel (hierfür gibt es effiziente Verfahren). Analog kann man bei anderen kleinen e vorgehen. Man sollte e z. B. $2^{16} + 1$ wählen.

Die Moduln m_i aller Teilnehmer sollten verschieden sein.
Es werde eine Nachricht x verschlüsselt sowohl mit (m, e_1) als auch mit (m, e_2) und es gelte $\operatorname{ggt}(e_1, e_2) = 1$. Dann kennt man die beiden Schlüsseltexte $y_i \equiv x^{e_i} \bmod m,\ i = 1, 2$. Hieraus kann man x berechnen:
Man bestimme Koeffizienten α_1, α_2 mit $\alpha_1 e_1 + \alpha_2 e_2 = 1$. Dann folgt

$$x = x^{\alpha_1 e_1 + \alpha_2 e_2} \equiv y_1^{\alpha_1} \cdot y_2^{\alpha_2} \bmod m.$$

Effizienz: Die Ver– und Entschlüsselung beim RSA erfordert eine Exponentation modulo m. Der Aufwand hängt von der Größe von e und d ab. Im allgemeinen ist e relativ klein, aber d hat dieselbe Größenordnung wie m, die Entschlüsselung dauert also länger. Die Entschlüsselung kann etwas beschleunigt werden:
Ist y der Schlüsseltext und d der private Schlüssel, so berechnet der Empfänger zunächst

$$x_p \equiv y^d \bmod p \quad \text{und} \quad x_q \equiv y^d \bmod q.$$

Dann werden Koeffizienten α_p und α_q bestimmt mit

$$\alpha_p \cdot p + \alpha_q \cdot q = 1.$$

Der Klartext ergibt sich dann (wie beim chinesischen Restsatz) durch

$$x \equiv (\alpha_q q \cdot x_p + \alpha_p p \cdot x_q) \bmod m.$$

Man beachte, daß die Zahlen $\alpha_q \cdot q$ und $\alpha_p \cdot p$ nicht von der Nachricht abhängen und nur einmal bestimmt werden müssen. Dieses Vorgehen beschleunigt die Entschlüsselung ungefähr um den Faktor 2.

RSA–Signaturen: Im Abschnitt 5.1 hatten wir vereinfachend angenommen, daß alle Blocklängen identisch sind. Hier haben wir jedoch verschiedene m_i der Benutzer. Will i an j die Signatur U schicken, so verwendet er im Falle $m_i < m_j$ die Abbildung

$$C_{m_j,e_j} \circ i_{m_i,m_j} \circ C_{m_i,d_i} \; : \; \mathbb{Z}/m_i \to \mathbb{Z}/m_j$$

und im Falle $m_i > m_j$ die Abbildung

$$C_{m_i,d_i} \circ i_{m_j,m_i} \circ C_{m_j,e_j} \; : \; \mathbb{Z}/m_j \to \mathbb{Z}/m_i.$$

In beiden Fällen kann dies auf die übliche Weise zu einer Abbildung $\mathcal{A}^l \to \mathcal{A}^L$ ergänzt werden.

5.3 Das Verfahren von Rabin

Auch das Rabin–Verfahren ist vom Typ $\mathbb{Z}/m \to \mathbb{Z}/m$.

(1) Wähle zwei verschiedene *große* 'Zufallsprimzahlen' p und q mit $p, q \equiv 3 \bmod 4$.
Setze $m := p \cdot q$.

(2) Sei $C_m : \mathbb{Z}/m \to \mathbb{Z}/m$ definiert durch $C_m(x) = x^2$.

(3) Bestimme $\alpha_p, \alpha_q \in \mathbb{Z}$ mit $\alpha_p \cdot p + \alpha_q \cdot q = 1$.

(4) Seien $D_m^{1,2,3,4} : \mathbb{Z}/m \to \mathbb{Z}/m$ definiert durch

$$D_m^{1,2,3,4}(y) = \pm\alpha_p p \cdot y^{(q+1)/4} \pm \alpha_q q \cdot y^{(p+1)/4}.$$

Der öffentliche Schlüssel ist m und (p,q) ist der geheime Schlüssel.

Bei der Entschlüsselung zieht man die Wurzel modulo m. Dabei sind $\pm y^{(p+1)/4}$ und $\pm y^{(q+1)/4}$ die Wurzeln modulo p bzw. q, die man zunächst bestimmt. Die 4 Wurzeln modulo m ergeben sich dann durch Anwendung des chinesischen Restsatzes. Den ursprünglichen Klartext erhält man durch Ausprobieren der 4 Möglichkeiten, nur eine davon wird sinnvoll in den Kontext passen. Es gibt aber auch die Möglichkeit, nur solche Klartexte zuzulassen, die eine gewisse Struktur haben (z. B. die ersten 64 Bits sind identisch mit den letzten 64 Bits). Dann ist das Wurzelziehen eindeutig.

Bemerkung: Die Einschränkung $p, q \equiv 3 \bmod 4$ wurde nur gemacht, damit das Wurzelziehen leichter fällt.

Lemma 5.5 *Es gilt $C_m(y) = x^2$ genau dann, wenn $y = D_m^i(x^2)$, für ein $i \in \{1,2,3,4\}$.*

Wegen der Nichteindeutigkeit des Wurzelziehens ist das Rabin–Verfahren streng genommen kein Kryptosystem. Es gilt $\mathcal{P} = \mathcal{C} = \mathbb{Z}/m$ und

$$\mathcal{K} = \{(m,p,q) \,:\, p,q \text{ Primzahlen mit } m = pq\}$$

und für $K = (m,p,q)$ ist

$$E_K(x) = x^2 \quad \text{und} \quad D_K^{1,2,3,4}(y) = D_m^{1,2,3,4}(y).$$

Beispiel: $p = 11,\ q = 23,\ m = 253$. $x = 158$ wird verschlüsselt zu $170 \equiv 158^2 \bmod 253$. Es gilt z. B. $\alpha_p = -2,\ \alpha_q = 1$. Zunächst sind die Wurzeln modulo p und q: $170^{(p+1)/4} = 170^3 \equiv 5^3 \equiv 4 \bmod 11$ und $170^{(q+1)/4} = 170^6 \equiv 9^6 \equiv 3 \bmod 23$. Es folgt $D_m^i(179) = \pm 2 \cdot 11 \cdot 3 \pm 1 \cdot 23 \cdot 4$, also $26, 95, 158, 227$.

Bemerkung: (1) Der Aufwand zum Entschlüsseln beim Rabin Verfahren entspricht dem beim RSA (in der modifizierten Version). Zum Verschlüsseln ist aber nur eine Quadrierung nötig. Hier ist also das Rabin Vefahren effizienter als RSA.

(2) Für die Wahl von p und q gelten die gleichen Bemerkungen wie bei RSA.

(3) Analog zu RSA bei kleinem Exponenten e ist eine Attacke möglich. Als Gegenmaßnahme kann man einige Nachrichtenbits zufällig wählen.

(4) Im Gegensatz zu RSA ist bekannt, daß das Brechen des Rabin Verfahrens äquivalent zur Faktorisierung von m ist.

Einerseits kann man das Rabin Vefahren natürlich brechen, wenn man m faktorisieren kann. Sei andererseits angenommen, es gebe einen Algorithmus $\mathcal{R}$, mit dem man das Rabin Verfahren brechen kann, d. h. zu jedem $y = x^2 \in \mathbb{Z}/m$ bestimmt $\mathcal{R}$ eine Quadratwurzel $x = \mathcal{R}(y)$. Mit Hilfe von $\mathcal{R}$ kann man dann m faktorisieren: Man wähle ein zufälliges $a \in \{1, \ldots, m-1\}$. Falls $\text{ggt}\,(a,m) \neq 1$, hat man einen Teiler gefunden. Ansonsten berechnet man $y \equiv a^2 \bmod m$ und $b = \mathcal{R}(y)$. Dann gilt $b \equiv \pm a \bmod p$ und $b \equiv \pm a \bmod q$. In der Hälfte aller Fälle sind die Vorzeichen verschieden und es gilt $\text{ggt}\,(b-a,m) = p$ bzw. $\text{ggt}\,(b-a,m) = q$. Damit hat man mit Wahrscheinlichkeit $1/2$ einen Teiler gefunden. Durch Wiederholung findet man mit großer Wahrscheinlichkeit schnell einen Teiler.

5.4 Das Verfahren von ElGamal

Das Verfahren nach ElGamal beruht auf der Schwierigkeit, in einer endlichen abelschen Gruppe G diskrete Logarithmen zu bestimmen. Dies trifft nach heutiger Kenntnis insbesondere auf die Einheitengruppe $\mathbb{F}_q^* = (\mathbb{F}_q \setminus \{0\}, \cdot)$ eines endlichen Körpers $\mathbb{F}_q$ zu. Man kann aber die gleichen Überlegungen für beliebige zyklische Gruppen mit dieser Eigenschaft anstellen. Dabei sollte nur gewährleistet sein, daß sich Schlüsselerzeugung, Ver– und Entschlüsselung effizient implementieren läßt, z. B.

- Die Punktgruppe einer elliptischen Kurve über einem endlichen Körper.
- Die Jakobische Varietät hyperelliptischer Kurven über endlichen Körpern.
- Die Klassengruppe imaginär quadratischer Ordnungen.

Beispiel: Ungeeignet ist z. B. $G = (\mathbb{Z}/m, +)$. Hier entspricht g^a der a–fachen Summe $g + \ldots + g = a \cdot g$ und daraus läßt sich das a leicht berechnen (Multiplikation mit g^{-1}).

Diffie–Hellmann Schlüsselaustausch: Ein Vorläufer des ElGamal Verfahrens ist das Verfahren von Diffie und Hellman zum Austausch eines Schlüssels (z. B. für ein symmetrisches Verfahren). Zwei Personen S und E können wie folgt über eine unsichere Leitung einen geheimen Schlüssel bestimmen: Dazu sei eine zyklische Gruppe G der Ordnung n vorgegeben.

(1) S wählt einen Erzeuger g von G (etwa mit Algorithmus 3.5).

(2) S wählt ein zufälliges $a \in \{1, \ldots, n-1\}$.

(3) S berechnet $A = g^a \in G$ und sendet g und A an E.

(4) E wählt ein zufälliges $b \in \{1, \ldots, n-1\}$.

(5) E berechnet $B = g^b$ und schickt B an S.

(6) S berechnet $B^a = g^{ab}$ und E berechnet $A^b = g^{ab}$ ($K = g^{ab}$ ist dann der geheime Schlüssel).

Bei Abhören der Leitung kennt man g, $A = g^a$ und $B = g^b$. Hieraus müsste der Schlüssel $K = g^{ab}$ berechnet werden. Kann man effizient diskrete Logarithmen bestimmen, so berechnet man a aus A oder b aus B und bestimmt K. Theoretisch könnte es aber einen anderen Weg geben, g^{ab} aus g^a und g^b zu bestimmen. Wir betrachten aber nur Gruppen, für die es (bisher) keinen effizienten Algorithmus hierfür gibt. Die **Diffie–Hellman Voraussetzung** besagt, daß dies für $\mathbb{F}_q^*$ der Fall ist.

Beispiel: $G = \mathbb{F}_{17}^* = (\mathbb{Z}/17\setminus\{0\}, \cdot)$, $g = 3$ ist ein Erzeuger. Mit $a = 7$ ist $A \equiv 3^7 \bmod 17$, also $A = 11$. Mit $b = 4$ ist $B \equiv 3^4 \bmod 17$, also $B = 13$.
S berechnet $B^a = 13^7 \bmod 17$, also $K = 4$. E berechnet $A^b = 11^4 \bmod 17$, also $K = 4$.

Beim ElGamal Verfahren geht man nun genau wie beim Diffie–Hellman Schlüsseltausch vor. Zunächst wählt man einen sehr großen endlichen Körper $\mathbb{F}_q$ und einen Erzeuger $g \in \mathbb{F}_q^*$. Wir gehen davon aus, daß sich die Textblöcke und die Elemente von $\mathbb{F}_q^*$ bijektiv entsprechen.

(1) Wähle ein zufälliges $a \in \{1, \ldots, q-2\}$ (a ist der geheime Schlüssel zum Entschlüsseln) und bestimme $A = g^a \in \mathbb{F}_q^*$ (A ist der öffentliche Schlüssel).

(2) Zu einem zufällig gewählten $k \in \{0, \ldots, q-1\}$ sei $C_{q,A,k} : \mathbb{F}_q^* \to \mathbb{F}_q^* \times \mathbb{F}_q^*$ definiert durch

$$C_{q,g,A,k}(x) = (g^k, x \cdot A^k).$$

(3) Sei $D_{q,a} : \mathbb{F}_q^* \times \mathbb{F}_q^* \to \mathbb{F}_q^*$ definiert durch

$$D_{q,a}(y_1, y_2) = y_1^{q-1-a} \cdot y_2.$$

Lemma 5.6 *$D_{q,a}(C_{q,g,A,k}(x)) = x$, für alle $x \in \mathbb{F}_q^*$ und alle $k \in \mathbb{N}$.*

Jeder Teilnehmer wählt einen geheimen Schlüssel a_i und gibt $A_i = g^{a_i}$ öffentlich bekannt. Will nun Teilnehmer i an Teilnehmer j eine geheime Nachricht x senden, so wählt er eine natürliche Zahl k zufällig und schickt x mit einer 'Maske' A_j^k (öffentlich bekannt) zusammen mit dem Anhang g^k. Teilnehmer j kennt a_j und kann x mittels D_{q,a_j} rekonstruieren.

Bemerkung: Das zufällig gewählte k des Senders entspricht dem b beim Diffie–Hellman Schlüsselaustausch.

Formal ist $\mathcal{P} = \mathbb{F}_q^*$, $\mathcal{C} = \mathbb{F}_q^* \times \mathbb{F}_q^*$,

$$\mathcal{K} = \{(q, g, a, A) : g \in \mathbb{F}_q \text{ primitives Element}, \ g^a = A\}.$$

Für $K = (q, g, a, A)$ und $k \in \mathbb{N}$ ist

$$E_K(x) = (g^k, x \cdot A^k) \quad \text{und} \quad D_K(y_1, y_2) = y_1^{q-1-a} \cdot y_2 \quad (= (y_1^a)^{-1} \cdot y_2).$$

Beispiel: $q = 16$ (vgl. das Beispiel am Ende von Kapitel 3). Erzeuger ist α (entspricht dem Polynom x oder auch $(0,1,0,0)$). Mit $a = 3$ ist $A = \alpha^3 = (0,0,0,1)$. Es sei $k = 4$ gewählt und der Klartext sei $x = (0,1,1,0)$. Gesendet wird $((0,1,0,0)^4, (0,1,1,0) \cdot (0,0,0,1)^4)$ (das ist ausgerechnet $((1,1,0,0),(0,0,1,0))$). Zum Entschlüsseln wird dann $(1,1,0,0)^{12} \cdot (0,0,1,0) = (0,1,1,0) = x$ berechnet.

Bemerkung: (1) Die Verschlüsselung benötigt zwei Exponentationen in $\mathbb{F}_q^*$. Die Entschlüsselung erfordert eine solche Exponentation.

(2) Ein Nachteil des ElGamal Verfahrens ist, daß der Schlüsseltext doppelt so lang ist wie der Klartext.

(3) q und g können für alle Benutzer gleich gewählt werden. Das erhöht aber das Risiko.

(4) q sollte groß genug sein (mindestens 512 Bits lang). Ferner sollte q so gewählt sein, daß bekannte Verfahren zur Berechnung diskreter Logarithmen in $\mathbb{F}_q^*$ nicht schnell zum Ziel kommen. Am besten wird q zufällig gewählt.

(5) Bei jeder ElGamal Verschlüsselung sollte der Exponent k neu gewählt werden. Verschlüsselt man x und x' mit dem gleichen k, so läßt sich aus der Kenntnis von x auch x' bestimmen: Kennt man die Schlüsseltexte $y = xA^k$ und $y' = x'A^k$, so gilt $x' = y' \cdot y^{-1} \cdot x$.
(6) Die Verschlüsselung im ElGamal Verfahren ist randomisiert. D. h. derselbe Klartext liefert bei wiederholter Verschlüsselung unterschiedliche Schlüsseltexte. Dies erhöht die Sicherheit.

DSS: Auf der Grundlage des ElGamal Verfahrens ist 1991 ein Signaturschema standardisiert worden, nämlich Digital Signature Standard (DSS).

(1) Wähle eine zufällige Primzahl q (Länge ca. 160 Bits).

(2) Wähle eine Primzahl p mit $p \equiv 1 \bmod q$ (Länge ca. 512 Bits).

(3) Wähle ein Element $g \in \mathbb{F}_p^*$ mit Ordnung q (ist erfüllt, wenn $g^q \equiv 1 \bmod p$).

(4) Wähle zufälliges $a \in \{1, \ldots, q-1\}$ (a ist der geheime Schlüssel) und berechne $A = g^a \bmod p$ (A ist öffentlicher Schlüssel).

Will nun ein Sender S eine Nachricht signieren, so wendet er zunächst eine geeignete Hashfunktion auf den Klartext an und erhält eine Zahl $h \in \{1, \ldots, q-1\}$. Dann wählt er eine zufällige Zahl $k \in \{1, \ldots, q-1\}$ und berechnet zuerst $v = g^k \bmod p$ und dann $r = v \bmod q$ (jeweils $v \in \{0, \ldots, p-1\}$ und $r \in \{0, \ldots, q-1\}$). Dann bestimmt S eine Zahl s mit

$$sk \equiv h + ar \bmod q.$$

Die Signatur ist dann das Paar (r, s). Um die Unterschrift zu verifizieren, berechnet der Empfänger E

$$u_1 = s^{-1}h \bmod q \quad \text{und} \quad u_2 = s^{-1}r \bmod q.$$

Danach berechnet E $g^{u_1} \cdot A^{u_2} \bmod p$. Ist das Ergebnis kongruent r modulo q, so ist E zufrieden. Man beachte, daß wegen $\operatorname{ord}(g) = q$ gilt

$$g^{u_1} \cdot A^{u_2} \equiv g^{s^{-1}h} \cdot g^{s^{-1}ar} \equiv g^{s^{-1}(h+ar)} \equiv g^k \equiv v \bmod p$$

und zur Bestimmung von s (solange das Verfahren nicht gebrochen ist) die Kenntnis von a nötig war.

Bemerkung: (1) Ein Vorteil von DSS ist die relativ kurze Signatur (ca. 2 mal 160 Bits).
(2) Die Sicherheit von DSS basiert auf dem diskreten Logarithmen Problem in $\mathbb{F}_p^*$ und p ist groß. (Es hat sich in der Praxis gezeigt, daß das Problem der diskreten Logarithmen in der von g erzeugten Untergruppe nicht leichter ist als in ganz $\mathbb{F}_p^*$).
(3) DSS ist ein sicheres, effizientes Signaturverfahren, das wenig Speicherplatz benötigt.

5.5 Rucksack–Verfahren

Das folgende Rucksack–Verfahren stammt von Merkle und Hellman (1978). Es beruht auf der Schwierigkeit des Teilmengen–Summen Problems.

Definition 5.7 *Gegeben sei eine Folge* $s_1, \ldots, s_n$ *von natürlichen Zahlen und ein weiteres* $T \in \mathbb{N}$. *Das* **Teilmengen–Summen Problem** *besteht darin, zu entscheiden, ob es einen Vektor* $x \in \{0,1\}^n$ *gibt mit* $s \cdot x = \sum_{i=1}^{n} x_i s_i = T$.

Die Frage ist also, ob man einen Rucksack mit Kapazität T mit einer Auswahl von Stücken der Größe s_i genau auffüllen kann. Der Name Rucksack–Verfahren ist eigentlich eine falsche Bezeichnung, da bei dem Rucksack Problem $\sum x_i s_i \leq T$ erlaubt ist und der Wert $\sum x_i c_i$ zu maximieren ist.

Das Teilmengen–Summen Problem ist als *NP–vollständig* bekannt. D. h. es gehört zu einer Klasse von Problemen (in der z. B. auch das Problem des Handlungsreisenden enthalten ist) von äquivalenter Komplexität, für die es vermutlich kein polynomiales Verfahren gibt. Beschränkt man sich aber auf eine spezielle Klasse von Folgen s, so kann man ein sehr effizientes Verfahren angeben.

Definition 5.8 *Eine Folge* $s_1, \ldots, s_n$ *von natürlichen Zahlen heißt* **überansteigend**, *wenn für* $j = 2, \ldots, n$ *gilt* $s_j > s_1 + \ldots + s_{j-1}$.

Beispiel: $s = (2, 5, 9, 21, 45, 103, 215, 450, 946)$ ist überansteigend.

Algorithmus 5.9 (Rucksack)
Eingabe: Eine überansteigende Folge $s = (s_1, \ldots, s_n)$ und $T \in \mathbb{N}$.
Ausgabe: Ein x mit $s \cdot x = T$ oder Antwort 'nicht lösbar'.

```
begin
  for i := n downto 1 do
  begin
    if T ≥ s_i then
    begin
      T := T − s_i;
      x_i := 1;
    end else x_i := 0;
  end;
  if T = 0 then x ist eindeutige Lösung else 'nicht lösbar';
end
```

Beispiel: Wie oben sei $n = 9$ und $s = (2, 5, 9, 21, 45, 103, 215, 450, 946)$. Ferner sei $T = 1643$. Algorithmus 5.9 liefert $x = (1, 0, 1, 1, 0, 0, 1, 1, 1)$.

Lemma 5.10 *Algorithmus 5.9 löst das Teilmengen–Summen Problem für überansteigende Folgen in einer Zeit von $O(n)$. Die Lösung x ist eindeutig bestimmt.*

Das Prinzip des folgenden Verfahrens von Merkle–Hellman ist, daß mit Kenntnis des privaten Schlüssels ein Teilmengen–Summen Problem für eine überansteigende Folge zu lösen ist. Ein Angreifer ohne diese Kenntnis müsste jedoch ein solches Problem lösen, das nicht diese spezielle Struktur hat.

(1) Wähle (zufällig) eine überansteigende Folge $s = (s_1, \ldots, s_n)$, eine Primzahl $p > s_1 + \ldots + s_n$ und eine Zahl $a \in \{1, \ldots, p-1\}$.

(2) Bestimme $b = a^{-1} \bmod p$.

(3) Bestimme $t = (t_1, \ldots, t_n)$ mit $t_i = a \cdot s_i \bmod p,\ i = 1, \ldots n$.

(4) Sei $E_t : \{0,1\}^n \to \{0, \ldots, n(p-1)\}$ definiert durch

$$E_t(x_1, \ldots, x_n) := \sum_{i=1}^{n} x_i t_i.$$

(5) Die Funktion $D_{s,b,p} : \{0, \ldots, n(p-1)\} \to \{0,1\}^n$ sei definiert durch

$$D_{s,b,p}(y) := \text{ Rucksack } (s, by \bmod p).$$

Die Folge t ist i. a. nicht mehr überansteigend. Sie dient als öffentlicher Schlüssel zum Verschlüsseln. $s,\ p,\ a$ und b bleiben geheim und stellen den privaten Schlüssel zum Entschlüsseln dar.

Lemma 5.11 *Es gilt $D_{s,b,p} \circ E_t(x) = x$, für alle $x \in \{0,1\}^n$.*

Formal ist also das Verfahren von Merkle–Hellman gegeben durch
$\mathcal{P} = \{0,1\}^n,\ \mathcal{C} = \{0, \ldots, n(p-1)\}$ und

$$\mathcal{K} = \{(s,p,a,b,t) : s \text{ ü. a.},\ p > \sum s_i \text{ prim},\ a \in \{1, \ldots, p-1\},\ ab \equiv 1 \bmod p,\ t_i = as_i \bmod p\}$$

und für $K = (s,p,a,b,t)$ durch

$$E_K(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i t_i \quad \text{und} \quad D_K(y) = \text{ Rucksack } (s, by \bmod p).$$

Beispiel: Wie oben sei $n = 9$ und $s = (2, 5, 9, 21, 45, 103, 215, 450, 946)$.
Ferner sei $p = 2003$ und $a = 1289$. Dann ergibt sich $b = 317$ und
$t = (575, 436, 1586, 1030, 1921, 569, 721, 1183, 1570)$.
Will nun der Sender die Nachricht $x = (1, 0, 1, 1, 0, 0, 1, 1, 1)$ (in binärer Form) verschicken, so berechnet er $y = x \cdot t = 575 + 1586 + 1030 + 721 + 1183 + 1570 = 6665$.
Der Empfänger berechnet zunächst $by \bmod p = 317 \cdot 6665 \bmod 2003 = 1643$. Dann löst er das Teilmengen–Summen Problem zu s und 1643 mit Algorithmus 5.9 und erhält die ursprüngliche Nachricht $x = (1, 0, 1, 1, 0, 0, 1, 1, 1)$.

Bemerkung: Zwar ist die Folge t nicht überansteigend, doch wird es durch eine recht einfache Transformation aus einer solchen erhalten. 1982 fand Shamir ein Verfahren, das Rucksack Probleme von diesem Typ in polynomialer Zeit löst (auf der Grunlage eines ganzzahligen Optimierungsverfahrens von Lenstra). Daher kann das Verfahren von Merkle–Hellman nicht mehr als sicher betrachtet werden. Eine Möglichkeit dies zu umgehen ist, mehrere solche Transformationen anzuwenden. Doch bleibt es zweifelhaft, ob diese Modifikationen auf Dauer sicher sind.

Es soll nun ein Verfahren von Chor und Rivest vorgestellt werden, das auch auf der Schwierigkeit des Rucksack (Teilmengen–Summen) Problems basiert und bisher noch nicht gebrochen wurde.

(1) Wähle (zufällig) eine Primzahl p und ein $r \in \mathbb{N}$ (jeweils 2–3 stellig), so daß in $\mathbb{F}_q^* = \mathbb{F}_{p^r}^*$ diskrete Logarithmen leicht zu berechnen sind (z. B. $q - 1$ hat nur kleine Primfaktoren).

(2) Wähle ein irreduzibles Polynom $f(x) \in \mathbb{F}_p[x]$ vom Grad r (also $\mathbb{F}_q = \mathbb{F}_p[x]/(f(x))$).

(3) Wähle einen Erzeuger $g \in \mathbb{F}_q^*$ und ein $z \in \mathbb{N}$.

(4) Wähle $n < \min\{p, r\}$.

(5) Berechne $b_1, \ldots, b_n$ mit $g^{b_i} = x - i \ \in \mathbb{F}_q, \ i = 1, \ldots, n$.

(6) Wähle eine zufällige Permutation π von $\{1, \ldots, n\}$ und setze $t_i := b_{\pi(i)} + z \bmod q - 1$.

(7) Sei $E_t : \{0,1\}^n \to \mathbb{N} \times \mathbb{N}$ definiert durch

$$E_t(x_1, \ldots, x_n) := \left(\sum_{i=1}^{n} x_i t_i, \sum_{i=1}^{n} x_i \right).$$

(8) Die Funktion $D_{\pi,z} = (D^1_{\pi,z}, \ldots, D^n_{\pi,z}) : \mathbb{N} \times \mathbb{N} \to \{0,1\}^n$ sei definiert durch

$$D^i_{\pi,z}(y_1, y_2) := \begin{cases} 1 & , \text{ falls } \pi(i) \text{ Nullstelle von } g^{y_1 - z y_2} \\ 0 & , \text{ sonst} \end{cases}$$

Veröffentlicht werden n und $t = (t_1, \ldots, t_n)$ (öffentlicher Schlüssel zum Verschlüsseln). $q, f(x), g, z, b = (b_1, \ldots, b_n)$ und π bleiben geheim und stellen den privaten Schlüssel zum Entschlüsseln dar.

Lemma 5.12 *Es gilt* $D_{\pi,z} \circ E_t(x) = x$, *für alle* $x \in \{0,1\}^n$.

Formal ist also das Verfahren von Chor–Rivest gegeben durch
$\mathcal{P} = \{0,1\}^n$, $\mathcal{C} = \mathbb{N} \times \mathbb{N}$ und

$$\mathcal{K} = \{(q, f(x), g, z, b, \pi, t) : s \text{ wie oben definiert}\},$$

und für $K = (q, f(x), g, z, b, \pi, t)$ durch

$$E_K(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i t_i \quad \text{und} \quad D_K(y_1, y_2) = D_{\pi,z}(y_1, y_2).$$

6 Primzahltests

Für Public–Key–Kryptosysteme benötigt man häufig große zufällige Primzahlen. Die Existenz von hinreichend vielen Primzahlen ist gewährleistet z. B. durch das Bertrandsche Postulat, das besagt, daß zwischen einer Zahl n und $2n$ garantiert eine Primzahl liegt. Genauer wird die Häufigkeit von Primzahlen beschrieben durch den Primzahlsatz. Ist $\pi(x) = |\{p \in \mathcal{P} : p \leq x\}|$, so gilt $\pi(x) \sim x/\ln x$. Genauer gilt für $x \geq 59$ (ohne Beweis)

$$\frac{x}{\ln x}\left(1 + \frac{1}{2\ln x}\right) < \pi(x) < \frac{x}{\ln x}\left(1 + \frac{3}{2\ln x}\right).$$

Eine sichere Methode, um zu überprüfen, ob eine Zahl $n \in \mathbb{N}$ eine Primzahl ist, ist die **Probedivision**. Hierbei wird für alle Zahlen $\leq [\sqrt{n}]$ überprüft, ob sie n teilen. Selbst wenn man hierbei nur Primzahlen als Teiler zulässt, so zeigt der Primzahlsatz, daß mindestens $\Omega(\sqrt{n}/\log\sqrt{n})$ Probedivisionen nötig sind, um eine Primzahl nachzuweisen. Im RSA–Verfahren werden Primzahlen $> 10^{75}$ benutzt, darum müssen mindestens 10^{38} Probedivisionen durchgeführt werden. Dies ist aber in einer respektablen Zeit nicht möglich.
In der Praxis behilft man sich mit Verfahren, die mit hoher Wahrscheinlichkeit feststellen können, ob eine Zahl prim ist. Solche Verfahren heißen **Primzahltests**.
Zunächst wird eine zufällige ungerade Zahl (in binärer Darstellung mit fester Anzahl k von Bits) erzeugt, wobei das erste und das letzte Bit 1 gesetzt wird. Dann werden zunächst Probedivisionen mit allen Primzahlen unterhalb einer Schranke B durchgeführt. Danach wird dann ein Test angewandt, der eventuell einige Male wiederholt wird. Besteht eine Zahl all diese Tests, so gilt sie (mit großer Wahrscheinlichkeit) als Primzahl.

6.1 Der Fermat–Test

Der Fermat–Test beruht auf dem kleinen Satz von Fermat, der besagt, daß zu einer Primzahl n und jeder Zahl $a \in \mathbb{Z}$ mit $\operatorname{ggt}(a, n) = 1$ gilt:

$$a^{n-1} \equiv 1 \bmod n. \tag{6.1}$$

Man bestimmt also zufällig eine zu n teilerfremde Zahl a und überprüft, ob (6.1) gilt. Ist dies nicht der Fall, so weiß man, daß n keine Primzahl ist. Die Umkehrung gilt jedoch nicht.

Beispiel: 15 ist keine Primzahl, da $2^{14} \equiv 2^2 = 4 \bmod 15$ (aber z. B. $11^{14} \equiv 1 \bmod 15$).

Definition 6.1 *Eine zusammengesetzte Zahl n heißt* **Pseudo–Primzahl** *zur Basis a, falls (6.1) gilt. Ist n Pseudo–Primzahl zu allen a mit* $\operatorname{ggt}(a, n) = 1$, *so heißt n* **Carmichael–Zahl**.

Beispiel: (1) $n = 341 = 11 \cdot 31$. n ist Pseudo–Primzahl zur Basis 2, aber wegen $3^{340} \equiv 56 \bmod 341$ nicht zur Basis 3.
(2) $n = 561 = 3 \cdot 11 \cdot 17$ ist eine Carmichaelzahl.

Satz 6.2 *Eine ungerade zusammengesetzte Zahl $n \geq 3$ ist genau dann eine Carmichael–Zahl, wenn gilt*

(a) n ist quadratfrei, d. h. n enthält keinen mehrfachen Primfaktor.

(b) Für jeden Primfaktor $p|n$ gilt $(p-1)|(n-1)$.

Korollar 6.3 *Eine Carmichael–Zahl hat mindestens 3 verschiedene Primteiler.*

Bemerkung: Da es Pseudo–Primzahlen und Carmichael–Zahlen gibt, ist der Fermat–Test für die Praxis nicht besonders geeignet. Man kann sogar zeigen, daß es unendlich viele Carmichael–Zahlen gibt.

6.2 Der Solovay–Strassen–Test

Der Primzahltest von Solovay–Strassen beruht auf der Berechnung von Quadratischen Resten. Die folgende Verallgemeinerung des bekannten Legendre–Symbols $\left(\frac{a}{p}\right)$ erleichtert praktische Berechnungen erheblich.

Definition 6.4 *Es sei $m \in \mathbb{N}$ ungerade mit $m = p_1 \cdot \ldots \cdot p_r$ ($p_i = p_j$ möglich). Zu $a \in \mathbb{Z}$ definiert man das* **Jacobi–Symbol** *durch $\left(\frac{a}{1}\right) := 1$ und*

$$\left(\frac{a}{m}\right) = \left(\frac{a}{p_1}\right) \cdot \ldots \cdot \left(\frac{a}{p_r}\right).$$

Bemerkung: Im Falle $\left(\frac{a}{m}\right) = -1$ hat die Kongruenz $x^2 \equiv a \bmod m$ keine Lösung. Umgekehrt folgt aber aus $\left(\frac{a}{m}\right) = 1$ nicht, daß a quadratischer Rest modulo m ist, wie das Beispiel $a = 2$ und $m = 15$ zeigt:

$$\left(\frac{2}{15}\right) = \left(\frac{2}{3}\right) \cdot \left(\frac{2}{5}\right) = (-1) \cdot (-1) = 1,$$

aber 2 ist kein quadratischer Rest modulo 15.

Für das Jacobi–Symbol gilt auch das Reziprozitätsgesetz.

Satz 6.5 *Seien m, m_1, m_2 ungerade und $a, b \in \mathbb{Z}$. Dann gilt*

(a) $\left(\frac{ab}{m}\right) = \left(\frac{a}{m}\right) \cdot \left(\frac{b}{m}\right)$.

(b) $\left(\frac{a}{m_1 m_2}\right) = \left(\frac{a}{m_1}\right) \cdot \left(\frac{a}{m_2}\right)$.

(c) Aus $a \equiv b \bmod m$ folgt $\left(\frac{a}{m}\right) = \left(\frac{b}{m}\right)$.

(d) $\left(\frac{-1}{m}\right) = (-1)^{(m-1)/2}$.

(e) $\left(\frac{2}{m}\right) = (-1)^{(m^2-1)/8}$.

(f) $\left(\frac{m_1}{m_2}\right) \cdot \left(\frac{m_2}{m_1}\right) = (-1)^{(m_1-1)(m_2-1)/4}$.

Hiermit kann man sofort ein rekursives Verfahren zur Berechnung des Jacobi–Symbols angeben.

Algorithmus 6.6 (Jacobi)
Eingabe: $a \in \mathbb{Z},\ m \in \mathbb{N},\ m \geq 3$ ungerade.
Ausgabe: Das Jacobi–Symbol $\left(\frac{a}{m}\right)$.
begin
Division mit Rest: $a = qm + r$;
$a := r$;
if $a = 0$ **then** jacobi $:= 0$ **else**
if $a = 1$ **then** jacobi $:= 1$ **else**
begin
schreibe $a = 2^k \cdot r$ mit ungeradem r;
jacobi $:= (-1)^{k(m^2-1)/8} \cdot (-1)^{(r-1)(m-1)/4} \cdot$ jacobi $\left(\frac{m}{r}\right)$;
end;
end

Natürlich werden die Exponenten von (-1) jeweils nicht explizit berechnet. Das Verfahren ist wesentlich schneller als das, das nur auf dem Legendre Symbol beruht (vgl. Zahlentheorie I). Der Grund ist, daß hier keine Primzerlegungen mehr bestimmt werden müssen.

Bemerkung: Ist m ungerade, so gilt $\left(\frac{2}{m}\right) = 1$, wenn $m \equiv \pm 1 \bmod 8$ und -1, wenn $m \equiv 3, 5 \bmod 8$.

Beispiel: $a = 13567$ und $n = 37813$. Dann ist $\left(\frac{a}{n}\right) = \left(\frac{37813}{13567}\right) = \left(\frac{10679}{13567}\right) = -\left(\frac{13567}{10679}\right) = -\left(\frac{2888}{10679}\right) = -\left(\frac{2}{10679}\right)^3 \cdot \left(\frac{361}{10679}\right) = -\left(\frac{10679}{361}\right) = -\left(\frac{210}{361}\right) = -\left(\frac{2}{361}\right) \cdot \left(\frac{105}{361}\right) = -\left(\frac{105}{361}\right) = -\left(\frac{361}{105}\right) = -\left(\frac{46}{105}\right) = -\left(\frac{2}{105}\right) \cdot \left(\frac{23}{105}\right) = -\left(\frac{23}{105}\right) = -\left(\frac{105}{23}\right) = -\left(\frac{13}{23}\right) = -\left(\frac{23}{13}\right) = -\left(\frac{10}{13}\right) = -\left(\frac{2}{13}\right) \cdot \left(\frac{5}{13}\right) = \left(\frac{5}{13}\right) = \left(\frac{13}{5}\right) = \left(\frac{3}{5}\right) = \left(\frac{5}{3}\right) = \left(\frac{2}{3}\right) = -1$.

Satz 6.7 *Algorithmus 6.6 berechnet das Jacobi–Symbol in der Zeit* $O(\log a \cdot \log m)$.

Es gilt die folgende Umkehrung des Euler–Kriteriums:

Satz 6.8 *Sei* $n \geq 3$ *eine ungerade Zahl. Für jedes* $a \in \mathbb{Z}$ *mit* $\operatorname{ggt}(a, n) = 1$ *gelte*

$$a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \bmod n. \tag{6.2}$$

Dann ist n *eine Primzahl.*

Korollar 6.9 *Sei* $n \geq 3$ *eine ungerade zusammengesetzte Zahl und* A *die Menge aller* a *mit* $0 < a < n$, $\operatorname{ggt}(a, n) = 1$ *und*

$$a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \bmod n.$$

Dann gilt $|A| \leq \varphi(n)/2$.

Beispiel: $n = 15$, also $(n-1)/2 = 7$.

a	1	2	4	7	8	11	13	14
$a^7 \bmod 15$	1	8	4	13	2	11	7	14
$\left(\frac{a}{3}\right)$	1	-1	1	1	-1	-1	1	-1
$\left(\frac{a}{5}\right)$	1	-1	1	-1	-1	1	-1	1
$\left(\frac{a}{15}\right)$	1	1	1	-1	1	-1	-1	-1

Hier ist also $A = \{1, 14\}$ und damit $|A| = 2 \leq \varphi(15)/2 = 4$.

Wählt man zu $n \notin \mathcal{P}$ zufällig eine Zahl $a \in \{1, \ldots, n-1\}$, so ist die Wahrscheinlichkeit, daß (6.2) gilt, höchstens $1/2$. Wiederholt man den Test k–mal, so ist die Wahrscheinlichkeit, keinmal ein Element aus A getroffen zu haben, $1/2^k$. Wir erhalten so den folgenden Primzahltest.

Algorithmus 6.10 (Solovay–Strassen)
Eingabe: $n \in \mathbb{N}$, Anzahl k der Wiederholungen.
Ausgabe: Antwort 'n nicht prim' oder 'n ist prim' mit Wahrscheinlichkeit $\geq 1 - 2^{-k}$.

```
begin
  prim := TRUE;
  if n gerade then prim := FALSE else
  begin
    for i := 1 to k do
    begin
      bestimme zufälliges a ∈ {1, ..., n-1};
      berechne j := (a/n) mit Algorithmus 6.6;
      if j = 0 then prim := FALSE; STOP; else
      begin
        berechne p := a^((n-1)/2) mod n mit Algorithmus 2.7;
        if j ≢ p mod n then prim := FALSE; STOP;
      end;
    end;
  end;
end
```

6.3 Der Miller–Rabin–Test

Der Miller–Rabin–Test basiert auf einer Verschärfung des kleinen Satzes von Fermat. Zunächst aber ein weiteres Primzahlkriterium:

Satz 6.11 *Es sei $n \in \mathbb{N}$ mit $n \equiv 3 \bmod 4$. Ferner gelte für alle $a \in \mathbb{Z}$ mit $(a,n) = 1$*

$$a^{(n-1)/2} \equiv \pm 1 \bmod n.$$

Dann ist n eine Primzahl.

Ist aber $n \equiv 1 \bmod 4$, so reicht die Bedingung $a^{(n-1)/2} \equiv \pm 1$ nicht aus. Z. B. gilt für die Carmichael–Zahl $n = 1729$, daß $a^{(n-1)/2} \equiv 1$ für alle zu n teilerfremden a (vgl. Lemma 6.12). In der folgenden Verallgemeinerung von Satz 6.11 werden auch diese Fälle berücksichtigt. Als Hilfsmittel dient

Lemma 6.12 *Sei $n \geq 9$ eine ungerade zusammengesetzte Zahl und $p_1, \ldots, p_r$ die verschiedenen Primteiler von n. Für jeden Teiler t von $(n-1)$ ist $B_t = \{x \in \{1, \ldots, n-1\} : (x,n) = 1, x^t \equiv 1 \bmod n\}$ eine Untergruppe von $P(n)$ (wobei x mit seiner Restklasse modulo n identifiziert wird) und es gilt*

$$|B_t| = \prod_{i=1}^{r} (t, p_i - 1).$$

Satz 6.13 *(a) Sei $n > 9$ ungerade und $n - 1 = 2^t \cdot d$, mit ungeradem d. Für jede zu n teilerfremde Zahl a gelte*

$$a^d \equiv 1 \bmod n \quad \textit{oder} \quad \exists s \in \{0, \ldots, t-1\} : a^{2^s d} \equiv -1 \bmod n. \tag{6.3}$$

Dann ist n eine Primzahl.

(b) Ist umgekehrt n keine Primzahl, so gilt für die Menge A aller zu n teilerfremden a mit $0 < a < n$, die (6.3) erfüllen

$$|A| \leq \varphi(n)/4.$$

Korollar 6.14 *Sei n ungerade und zusammengesetzt. Dann ist die Anzahl der $a \in \{1, \ldots, n-1\}$ mit (6.3) höchstens $(n-1)/4$.*

Bemerkung: Diejenigen a mit $(a,n) = 1$ und $a \notin A$ werden auch *Zeugen gegen die Primalität von n* genannt.

Wählt man zu $n \notin \mathcal{P}$ zufällig eine Zahl $a \in \{1, \ldots, n-1\}$, so ist die Wahrscheinlichkeit, daß (6.3) gilt, höchstens $1/4$ (genauere Analysen haben gezeigt, daß für 'zufällige' große Zahlen die Wahrscheinlichkeit viel geringer ist). Wiederholt man den Test k–mal, so ist die Wahrscheinlichkeit, keinmal ein Element aus A getroffen zu haben, $1/4^k$. Analog zum Algorithmus 6.10 erhalten wir so den folgenden Primzahltest.

Algorithmus 6.15 (Miller–Rabin)

Eingabe: $n \in \mathbb{N}$, Anzahl k der Wiederholungen.
Ausgabe: Antwort 'n nicht prim' oder 'n ist prim' mit Wahrscheinlichkeit $\geq 1 - 4^{-k}$.

begin
  if n gerade **then** prim := FALSE **else**
  begin
    schreibe $n-1$ als $n-1 = 2^t \cdot d$;
    for $i := 1$ **to** k **do**
    begin
      bestimme zufälliges $a \in \{1, \ldots, n-1\}$;
      berechne $p := a^d \bmod n$ mit Algorithmus 2.7;
      prim := FALSE;
      if $p \equiv \pm 1 \bmod n$ **then** prim := TRUE **else**
      for $j := 1$ **to** $t-1$ **do**
      begin
        $p := p \cdot p \bmod n$;
        if $p \equiv -1 \bmod n$ **then** prim := TRUE;
      end;
    end;
  end

Beispiel: $n = 15$, also $n - 1 = 2 \cdot 7$. Eine zu 15 teilerfremde Zahl a ist genau dann Zeuge gegen die Primalität von 15, wenn $a^7 \not\equiv \pm 1 \bmod 15$.

a	1	2	4	7	8	11	13	14
$a^7 \bmod 15$	1	8	4	13	2	11	7	14
$a^{14} \bmod 15$	1	4	1	4	4	1	4	1

Der Tabelle entnimmt man, daß es 6 Zeugen gibt. Beim Fermat–Test widerlegen nur 4 Zahlen die Primalität von 15.

Satz 6.16 *Sei $n \in \mathbb{N}$ ungerade und $a \in \mathbb{Z}$ mit $(a, n) = 1$. Dann gilt (6.3) $\Rightarrow$ (6.2) $\Rightarrow$ (6.1).*

7 Faktorisierung

Wie wir gesehen haben, hängt die Sicherheit von RSA und dem Rabin Verfahren eng mit Schwierigkeit zusammen, natürliche Zahlen in ihre Primfaktoren zu zerlegen. Bisher gibt es hierfür keine effiziente Verfahren. Sollte in Zukunft ein solches gefunden werden, so sind beide Verfahren nicht mehr zu gebrauchen. Generell sollte man kryptographische Systeme so einrichten, daß die grundlegenden Verfahren leicht ersetzt werden können.
Hier sollen einige Faktorisierungalgorithmen beschrieben werden. Wir gehen davon aus, daß eine natürliche Zahl n gegeben ist, die bereits als zusammengesetzt erkannt wurde. Dies geht mit den Verfahren aus den vorigen Kapiteln sehr schnell, sie liefern aber keinen expliziten Faktor von n. Weiter reicht es aus, einen Teiler zu finden. Durch Wiederholung kann man n komplett faktorisieren.

7.1 Die Probedivision

Die naivste Methode ist es, einfach alle möglichen Teiler durchzuprobieren. Dies ist natürlich nur für kleine n praktisch möglich. Häufig speichert man alle Primzahlen unter einer festen Schranke B in einer Liste ab (gebräuchlich ist etwa $B = 10^6$). Für die Zahl n kann man dann alle kleinen Primteiler und die entsprechenden Vielfachheiten bestimmen.

Beispiel: $n = 3^{21} + 1 = 10460353204$. Probedivision durch alle Primzahlen bis 50 ergibt die Faktoren 2^2, 7^2 und 43. Übrig bleibt 1241143.

7.2 Die (p – 1)–Methode

Die folgende Methode stammt von John Pollard. Sie beruht auf der folgenden Tatsache. Ist p ein Primteiler von n und $\operatorname{ggt}(a, n) = 1$, dann gilt $a^k \equiv 1 \bmod p$ für alle Vielfachen k von $p-1$. Ist zusätzlich $a^k \not\equiv 1 \bmod n$, so ist $d = \operatorname{ggt}(a^k - 1, n)$ ein echter Teiler von n (vgl. den Begriff des Teilererzeugers in Abschnitt 5.2).
Das Problem ist, daß p ja gerade nicht bekannt ist. Um dennoch ein geeignetes k zu finden, setzt man k gleich dem Produkt aller Primzahlpotenzen p^α, die unterhalb einer Schranke B liegen, also

$$k := \prod_p p^{[\log_p(B)]}. \tag{7.1}$$

Hat nun n einen Primteiler p, so daß alle Primzahlpotenzen in $(p-1)$ unter B liegen, so ist k ein Vielfaches von $(p-1)$. Findet man ein a mit den obigen Eigenschaften, so ist das entsprechende d ein echter Teiler von n.

Algorithmus 7.1 (pminus1)
Eingabe: $n \in \mathbb{N}$ nicht prim.
Ausgabe: Evtl. ein Teiler von n.

begin
  Bestimme ein zufälliges $a \in \{2, \ldots, n-1\}$;
  $d :=$ ggt (a, n);
  if $d > 1$ **then** teiler $:= d$ **else**
  begin
    berechne $b = a^k \bmod n$;
    if $b = 1$ **then** 'Kein Teiler' **else**
    begin
      $d :=$ ggt $(b-1, n)$;
      if $d > 1$ **then** teiler $:= d$ **else** 'Kein Teiler'
    end;
  end;
end

Bemerkung: (1) Falls kein Teiler gefunden wird, so kann man B auf B' erhöhen. Dabei kann man aus dem alten k das k' durch Heranmultiplizieren der fehlenden Faktoren erhalten (und aus b durch Potenzieren mit k'/k das b'), usw.
(2) Die 'meisten' Primzahlen haben die geforderte Eigenschaft nicht.

Beispiel: $n = 1241143$, $a = 2$ und $B = 15$. Hier ist $k = 2^3 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 = 360360$. Ferner ist $b = 861526$ Es gilt $d =$ ggt $(861525, 1241143) = 547$. Damit ist also $p = 547$ ein Primteiler. Der Kofaktor $q = 2269$ ist auch eine Primzahl. Die Zerlegung von n ist also $n = 547 \cdot 2269$. (Man beachte: $546 = 2 \cdot 3 \cdot 7 \cdot 13$.)

7.3 Die Pollardsche ϱ–Methode

Es sei $n \in \mathbb{N}$ mit den Primteilern $p_1, \ldots, p_r$. Weiter seien $x, y \in \mathbb{N}$ gegeben mit $x \not\equiv y \bmod n$. Es kann aber sein, daß $x \equiv y \bmod p_i$ für ein $i \in \{1, \ldots, r\}$. Ohne die p_i zu kennen, kann man dies prüfen, indem man $d =$ ggt $(x - y, n)$ berechnet. Ist $d > 1$, so ist es ein Produkt der p_i.
Die Idee ist nun, so viele $x_1, \ldots, x_k$ zufällig aus $\{0, \ldots, n-1\}$ zu wählen, daß nach den Gesetzen der Wahrscheinlichkeit zu erwarten ist, daß $x_\nu \equiv x_\mu \bmod p_i$ für wenigstens ein Paar ν, μ. Dazu stellen wir zunächst eine kombinatorische Vorüberlegung an.

Satz 7.2 *Sei M eine Menge mit m Elementen. Aus M werden $k \leq m$ Stichproben mit Zurücklegen genommen. Dann ist die Wahrscheinlichkeit dafür, daß alle Stichproben paarweise voneinander verschieden sind, gleich*

$$P_{m,k} = \prod_{i=1}^{k-1} \left(1 - \frac{i}{m}\right).$$

Bemerkung: (1) Mit $m = 365$ und $k = 30$ entspricht dies dem **Geburtstagsparadoxon**. In einer zufälligen Gesellschaft von 30 Personen ist die Wahrscheinlichkeit, daß keine 2 am gleichen Tag Geburtstag haben, gleich $P_{m,k} = 0.2936\ldots$, also weniger als 30 Prozent (bei 50 Personen sogar weniger als 3 Prozent).
(2) Es gilt $P_{m,k} \sim \exp(-k(k-1)/(2m))$. Der kritische Wert 1/2 wird also bei ungefähr $k \sim 1.2\sqrt{m}$ erreicht.

Auf unser Problem angewendet heißt das: Ist p ein Primteiler von n und sind $x_1, \ldots, x_k$ aus $\{0, \ldots, n-1\}$ zufällig gewählt, wobei $k = c\sqrt{p}$, mit einer kleinen Konstanten $c > 1$, so können wir damit rechnen, daß zwei der Zahlen kongruent modulo p sind. Allerdings ist der Aufwand, für alle Paare ν, μ den ggt $(x_\nu - x_\mu, n)$ zu berechnen, zu groß. Stattdessen benutzt man den folgenden Trick: Die x_i werden als Pseudo–Zufallsfolge rekursiv definiert durch $x_{i+1} := f(x_i)$, mit einer (geeignet gewählten) Funktion $f : \mathbb{Z}/n \to \mathbb{Z}/n$ und einem Startwert x_0. Gilt einmal $x_\nu \equiv x_\mu \bmod p_j$, so dauert es nicht allzu lange bis $x_i \equiv x_{2i} \bmod p_j$ für ein i.

Satz 7.3 *Sei M endliche Menge und $f : M \to M$. Für einen Anfangswert $x_0 \in M$ sei die Folge $\{x_i\}$ rekursiv definiert durch $x_{i+1} := f(x_i)$. Dann gibt es ganze Zahlen $n_0 \geq 0$, $k > 0$, so daß*

$$x_{i+k} = x_i, \quad \textit{für alle } i \geq n_0.$$

Es seien n_0 und k minimal gewählt (k heißt dann Periode und n_0 Vorperiode). Ist $y_i = x_{2i}$, dann gibt es ein minimales $i_0 > 0$ mit $x_{i_0} = y_{i_0}$. Es gilt $k | i_0$ und falls $n_0 \leq k$ sogar $i_0 = k$. In jedem Fall ist i_0 das kleinste Vielfache von k mit $i_0 \geq n_0$.

Ist die Periode groß, so kann man nicht alle auftretenden x_ν speichern. Man berechnet stattdessen parallel $x_{i+1} = f(x_i)$ und $y_{i+1} = f(f(y_i))$ (mit $y_0 = x_0$) und speichert nur x_i und y_i ab.
Es bleibt die Funktion $f : \mathbb{Z}/n \to \mathbb{Z}/n$ zu spezifizieren. In der Praxis hat sich die Funktion

$$f(x) := x^2 + a \bmod n$$

mit einer Konstanten $a \notin \{0, -2\}$ besonders bewährt. Es ergibt sich der folgende Algorithmus:

Algorithmus 7.4 (pollrho)

Eingabe: $n \in \mathbb{N}$ nicht prim.
Ausgabe: Evtl. ein Teiler von n.

begin
  Bestimme ein zufälliges $x \in \{1, \ldots, n-1\}$;
  Setze $y := x$;
  for $i := 1$ **to** anzahl **do**
  begin
    $x := f(x)$;
    $y := f(f(y))$;
    $d :=$ ggt $(x - y, n)$;
    if $1 < d < n$ **then** teiler $:= d$; STOP;
  end;
  'Kein Teiler';
end

Bemerkung: (1) Das Verfahren kann aus zwei Gründen scheitern. Einmal kann jeweils $d = 1$ sein. Dann sind wahrscheinlich die Primfaktoren p von n so groß, daß noch kein Zyklus in $\mathbb{Z}/p$ entstehen konnte. Die zweite Möglichkeit ist, daß einmal $y_i \equiv x_i \bmod n$, also $d = n$ wird. In diesem Falle sollte man mit einem neuen Startwert beginnen.
(2) Der Name des Algorithmus stammt von der 'Bahn' der x_i, die wie ein ϱ aussieht (Vorperiode + Zykel).
(3) Man kann die Laufzeit verkürzen, indem man jeweils mehrere (x_i, y_i) zusammenfaßt und $d =$ ggt $(\prod_i (x_i - y_i), n)$ berechnet.

Beispiel: $n = 4087$, $x_0 = 8$ und $f(x) = x^2 + 2$. Es ergibt sich

$$x_1 = 66,\ x_2 = 271,\ x_3 = 3964,\ x_4 = 2870,\ x_5 = 1597,\ x_6 = 123,\ x_7 = 2870,\ x_8 = 1597, \ldots$$

und

$$y_1 = 271,\ y_2 = 2870,\ y_3 = 123,\ y_4 = 1597,\ y_5 = 2870,\ y_6 = 123,\ y_7 = 1597,\ y_8 = 2870, \ldots$$

Damit ist die Periode 3 und die Vorperiode ist 4. Es gilt $i_0 = 6$. Für die größten gemeinsamen Teiler gilt

$$d_1 = 1,\ d_2 = 1,\ d_3 = 1,\ d_4 = 67,\ d_5 = 67,\ d_6 = 4087,\ d_7 = 67,\ d_8 = 67,\ d_9 = 4087, \ldots$$

Man erhält den Teiler 67 und damit $n = 61 \cdot 67$. Modulo $p = 67$ gilt

$$x_1 = 66,\ x_2 = 3,\ x_3 = 11,\ x_4 = 56,\ x_5 = 56, \ldots$$

Hier ist die Periode also sogar 1 und die Vorperiode 4.

7.4 Fermat Faktorisierung und Faktorbasen

Die **Fermat Faktorisierung** benutzt die Tatsache, daß man $n = ab$ auch schreiben kann als

$$n = t^2 - s^2 = (t+s)\cdot(t-s) \quad \text{mit } t = \frac{a+b}{2} \text{ und } s = \frac{a-b}{2}.$$

Hat man umgekehrt eine solche Darstellung von n gefunden, so kann man die Faktoren $a = t+s$ und $b = t-s$ bestimmen.
Die Fermat Faktorisierung führt zum Ziel, wenn $n = ab$ mit Zahlen a und b, die nicht weit auseinander liegen. Man startet dabei mit $t = [\sqrt{n}]+1, \ldots$ und prüft, ob t^2-n eine Quadratzahl s^2 ist.

Beispiel: $n = 200819$, $[\sqrt{n}] + 1 = 449$.
$t = 449$: $449^2 - n = 782$ ist keine Quadratzahl.
$t = 450$: $450^2 - n = 1681 = 41^2$.
Also ist $n = 450^2 - 41^2 = (450+41)\cdot(450-41) = 491 \cdot 409$.

Im Allgemeinen muß man sehr viele t ausprobieren, bis man zum Ziel kommt. Manchmal hilft die folgende Verallgemeinerung der Fermat Faktorisierung.
Wähle ein kleines $k \in \mathbb{N}$ und teste für $t = [\sqrt{kn}] + 1, \ldots$ ob $t^2 - kn$ eine Quadratzahl s^2 ist. Im positiven Fall gilt $t^2 - s^2 = (t+s)\cdot(t-s) = kn$ und $\text{ggt}\,(t+s, n) > 1$ und $\text{ggt}\,(t-s, n) > 1$ sind echte Teiler von n.

Beispiel: $n = 141467$.
$k = 1$: $t = 377, 378, \ldots$ dauert zu lange.
$k = 3$: $t = [\sqrt{3n}] + 1 = 652, \ldots$, bei 655: $655^2 - 3n = 68^2$. Es gilt $\text{ggt}\,(655 + 68, n) = 241$ und damit $n = 241 \cdot 587$ ($k = 3$ funktionierte, weil $587 \sim 3 \cdot 241$).

Eine weitere Verallgemeinerung der Idee, die hinter der Fermat Faktorisierung steckt, beruht auf dem folgenden Lemma.

Lemma 7.5 *Sei $n \in \mathbb{N}$ und $t, s \in \mathbb{Z}$ mit*

$$t^2 \equiv s^2 \bmod n, \quad \textit{aber} \quad t \not\equiv \pm s \bmod n.$$

Dann sind $d_1 = \text{ggt}\,(t+s, n)$ und $d_2 = \text{ggt}\,(t-s, n)$ echte Teiler von n.

Beispiel: $n = 4633$. Es gilt $118^2 \equiv 25 = 5^2 \bmod n$. Weiter ist $d_1 = (118 + 5, n) = 41$ und $d_2 = (118 - 5, n) = 113$ und man erhält $n = 41 \cdot 113$.

Die Frage ist nun: Wie findet man systematisch solche Zahlen wie 118 im Beispiel?
Die Idee ist es, mehrere solcher Zahlen b_i zu nehmen, deren absolut kleinste Reste modulo n (d. h. die ganze Zahl aus $[-n/2, n/2]$, die zu b_i kongruent ist, Bezeichnung: $A_n(b_i^2)$) Produkte von kleinen Primzahlen sind. Aus diesen kann man dann durch geschicktes Multiplizieren eine Quadratzahl s^2 erhalten.

Definition 7.6 *Eine* **Faktorbasis** *ist eine Menge $B = \{p_1, \ldots, p_r\}$, wobei die p_i verschiedene Primzahlen sind bis auf die Ausnahme, daß evtl. $p_1 = -1$. Ein $a \in \mathbb{Z}$ heißt* **B–Zahl**, *falls a sich als Produkt von Zahlen aus B schreiben läßt.*

Zu gegebener Faktorbasis B und $a \in \mathbb{Z}$ kann man leicht prüfen, ob a eine B–Zahl ist. Für alle Elemente aus B spaltet man möglichst viele Faktoren p von a ab. a ist genau dann eine B–Zahl, wenn eine 1 oder -1 übrigbleibt, nachdem alle $p \in B$ durchlaufen sind.

Beispiel: $n = 4633$, $B = \{-1, 2, 3\}$. Die Zahlen $A_n(67^2), A_n(68^2)$ und $A_n(69^2)$ sind B–Zahlen, da

$$A_n(67^2) = -144 = (-1) \cdot 2^4 \cdot 3^2, \quad A_n(68^2) = -9 = 3^2, \quad A_n(69^2) = 128 = 2^7.$$

Einer B–Zahl a kann man einen Vektor ϵ aus dem Vektorraum $(\mathbb{F}_2)^r$ wie folgt zuordnen. Ist $a = \prod_{i=1}^r p_i^{\alpha_i}$, dann setzt man

$$\epsilon_i = \begin{cases} 0, & \alpha_i \text{ gerade} \\ 1, & \alpha_i \text{ ungerade} \end{cases}$$

Beispiel: Im vorigen Beispiel sind die Vektoren zu 67, 68, 69 gleich $(1,0,0), (1,0,0)$ und $(0,1,0)$.

Gegeben sei eine Menge von Zahlen b_i, so daß die zugehörigen $a_i := A_n(b_i^2)$ B–Zahlen sind und die zugehörigen Vektoren ϵ_i addiert den Nullvektor ergeben (d. h. die ϵ_i sind linear abhängig; unter $r+1$ Vektoren findet man in jedem Fall eine solche Menge). Nach Voraussetzung kann man die a_i schreiben als $a_i = \prod_{j=1}^r p_j^{\alpha_{ij}}$. Setzt man nun

$$a := \prod_i a_i = \prod_{j=1}^r p_j^{\sum_i \alpha_{ij}},$$

so sind nach Voraussetzung die Exponenten aller p_j gerade, d. h. a ist Quadratzahl $a = c^2$ mit

$$c = \prod_{j=1}^r p_j^{\sum_i \alpha_{ij}/2}.$$

Mit $b := \prod_i b_i \bmod n$ (kleinste nichtnegative Reste) gilt dann $b^2 \equiv c^2 \bmod n$. Gilt $b \equiv \pm c \bmod n$, so sollte man von neuem beginnen (d. h. eine andere Teilmenge der b_i wählen oder neue b_i hinzunehmen oder die Faktorbasis geeignet verändern). Sind die b_i zufällig gewählt, so ist die Wahrscheinlichkeit hierfür nur $2/2^k$, wobei k die Anzahl der verschiedenen Primteiler von n ist.

Beispiel: In obigem Beispiel ist die Summe der zu 67 und 68 gehörenden Vektoren der Nullvektor. Man erhält

$$b = 67 \cdot 68 \equiv -77 \bmod n, \quad a = 144 \cdot 9 = 36^2, \quad \text{also } c = 36.$$

Es folgt $(-77)^2 \equiv 36^2 \bmod n$, aber (glücklicherweise) $-77 \not\equiv \pm 36 \bmod n$. Ein Faktor ist somit $\text{ggt}\,(-77 + 36, 4633) = 41$.

Wie wählt man nun die Faktorbasis B und geeignete b_i aus? Eine Möglichkeit ist es, B als -1 und die ersten $r-1$ Primzahlen zu wählen und dann zufällig nach den passenden b_i's zu suchen. Eine andere Möglichkeit ist es, zunächst einige b_i zu suchen, deren $A_n(b_i^2)$ einen kleinen Betrag haben und anschließend erst B geeignet zu wählen. Es gibt viele Varianten von Algorithmen zur Faktorisierung, die auf Faktorbasen aufbauen. Wir geben hier ein systematisches Verfahren.

Algorithmus 7.7 (faktorbasis)

Eingabe: $n \in \mathbb{N}$ nicht prim.
Ausgabe: Ein Teiler von n.

begin
  Wähle eine Zahl P; (z. B. 5–6 Stellen, wenn n 50–stellig);
  $B := \{-1\} \cup (\mathcal{P} \cap \{2, \ldots, P\})$;
  BZ $:= \emptyset$;
  repeat
    Wähle b zufällig;
    Berechne $a = A_n(b^2)$;
    if a ist eine B–Zahl **then** BZ := BZ $\cup\{b\}$;
  until card BZ = card B + 1;
  Bestimme die zu b_i gehörigen Vektoren ϵ_i;
  Bestimme eine Indexmenge I mit $\sum_{i\in I} \epsilon_i = 0$; (z. B. mit Zeilenreduktion)
  Bestimme a, $c = \sqrt{a}$ und $b = \prod_{i\in I} b_i$;
  if $b \equiv \pm c \bmod n$ **then** 'Neuer Versuch';
  else teiler := ggt $(b + c, n)$;
end

7.5 Das quadratische Sieb

In dem Verfahren im vorigen Abschnitt mußte für jedes einzelne b_i geprüft werden, ob $a_i = A_n(b_i^2)$ eine B–Zahl ist. Man kann die b_i aber wesentlich schneller finden, indem man wie bei dem Sieb des Erasthostenes vorgeht. Man hält sich eine Liste

$$T = \{[\sqrt{n}] + 1, \ldots, [\sqrt{n}] + A\},$$

mit einer geeignet gewählten ganzen Zahl A. Wie bei der Fermat Faktorisierung benutzt man die Werte

$$S = \{s(t) = t^2 - n \,:\, t \in T\}.$$

$s(t)$ ist genau dann durch p^α teilbar, wenn $t^2 \equiv n \bmod p^\alpha$ (insbesondere sollte n quadratischer Rest modulo p sein). Von dieser Kongruenz gibt es genau zwei Lösungen t_1, t_2 modulo p^α. Man ist dann sicher, daß genau für die $t \in T$ mit $t \equiv t_1, t_2 \bmod p^\alpha$ das $s(t)$ durch p^α teilbar ist. Nur diese (ausgesiebten) Zahlen werden dann durch p^α geteilt. Somit muß man keine erfolglosen Probedivisionen mehr durchführen. Die Lösungen der Kongruenz $t^2 \equiv n \bmod p^\alpha$ lassen sich sukzessive bestimmen (vgl. Zahlentheorie I).
Ist y eine Lösung von $t^2 \equiv n \bmod p^{\alpha-1}$ (für $p \neq 2$), so erhält man genau eine Lösung $z = y + x \cdot p^{\alpha-1}$ modulo p^α, wobei x eine Lösung der Kongruenz $(2y)x \equiv (n - y^2)/p^{\alpha-1} \bmod p$.

Im Fall $p = 2$ unterscheidet man zunächst einmal, ob $n \equiv 1 \bmod 4$. Trifft dies nicht zu, so entält $s(t)$ genau für die ungeraden t genau einen Faktor 2. Gilt $n \equiv 1 \bmod 4$ aber nicht $n \equiv 1 \bmod 8$ (also $n \equiv 5 \bmod 8$), dann entält $s(t)$ genau für die ungeraden t genau zwei Faktoren 2. Schließlich hat $s(t) \equiv 0 \bmod 2^\alpha$ für $s \geq 3$ genau 4 inkongruente Lösungen. Diese bestehen aus 2 Paaren. Ein Paar erzeugt die 4 Lösungen modulo $2^{\alpha+1}$, das andere keine.

Als Faktorbasis wählt man

$$B = \{2\} \cup \left\{ p \in \mathcal{P} \,:\, p \leq P,\ \left(\frac{n}{p}\right) = 1 \right\}.$$

In der Praxis bewährt hat sich eine Wahl der Zahlen A und P mit

$$A, P \sim e^{\sqrt{\log n \log\log n}}, \quad P < A < P^2. \tag{7.2}$$

Algorithmus 7.8 (quadratsieb)
Eingabe: $n \in \mathbb{N}$ ungerade und nicht prim.
Ausgabe: Eine Faktorbasis B und eine Menge BZ von Zahlen b_i mit $A_n(b_i^2)$ ist B–Zahl.

begin
  Wähle geeignete Zahlen A und P gemäß (7.2);
  $B := \{2\} \cup \{p \in \mathcal{P} \,:\, p \leq P,\ \left(\frac{n}{p}\right) = 1\}$;
  $T := \{[\sqrt{n}] + 1, \ldots, [\sqrt{n}] + A\}$;
  for $t := [\sqrt{n}] + 1$ **to** $[\sqrt{n}] + A$ **do** $s(t) := t^2 - n$;
  for $(t, p) \in T \times B$ **do** $M(t, p) := 0$;
  for $p \in B \backslash \{2\}$ **do** (Sieb mit $p \neq 2$)
  begin
    $\alpha := 0$;
    repeat
      $\alpha := \alpha + 1$;
      Bestimme Lösungen t_1, t_2 von $t^2 \equiv n \bmod p^\alpha$;
      for $t \in T$ mit $t \equiv t_1, t_2 \bmod p^\alpha$ **do**
      begin
        $M(t, p) := M(t, p) + 1$;
        $s(t) := s(t)/p$;
      end;
    until 'Keine Lösung in T';
  end;
  (** Sieb mit 2 **)
  if $n \not\equiv 1 \bmod 8$ **then**
  begin
    if $n \not\equiv 1 \bmod 4$ **then for** $t \in T$ ungerade **do** $M(t, 2) := 1$; $s(t) := s(t)/2$;
    else for $t \in T$ ungerade **do** $M(t, 2) := 2$; $s(t) := s(t)/4$;
  end else verfahre wie für $p \neq 2$ (beachte, daß es hier vier statt zwei Lösungen gibt);
  BZ $:= \{t \in T \,:\, s(t) = 1\}$;
end

Bemerkung: (1) Aus der Matrix M kann man sofort für alle $t \in$ BZ die Produktdarstellung angeben.
(2) Sind die Parameter A und P richtig gewählt, so ist die Menge BZ hinreichend groß. Man verfährt dann weiter wie im vorigen Abschnitt.
(3) Das quadratische Sieb war lange Zeit mit das effizienteste Faktorisierungsverfahren. Mittlerweile hat man ein schnellers Verfahren gefunden, das sogenannte **Zahlkörper–Sieb**.

Beispiel: $n = 4841$, $P = 45$ und $A = 16$. Nach prüfen der Jacobi–Symbole ergibt sich $B = \{2, 5, 7, 11, 17, 31, 43\}$. Für $t = 70, 71, \ldots, 85$ werden die Werte $s(t) = t^2 - n$ berechnet.

Sieb mit 5: Die Lösungen von $t^2 \equiv n \equiv 1 \bmod 5$ sind 1 und 4. Es werden also 71, 74, 76, 79, 81, 84 gesiebt.
Aus der Lösung $y = 1$ modulo 5 erhält man die Lösung $z = 21 = 1 + 5 \cdot 4$ modulo 25 ($x = 4$ ist Lösung von $(2 \cdot 1)x \equiv 4840/5 \equiv 3 \bmod 5$). Aus der Lösung $y = 4$ modulo 5 erhält man die Lösung $z = 4 = 4 + 5 \cdot 0$ modulo 25 ($x = 0$ ist Lösung von $(2 \cdot 4)x \equiv 4825/5 \equiv 0 \bmod 5$). Es werden also 71 und 79 gesiebt.
Analog erhält man modulo 125 die Lösungen 29 und 96. Hierzu ist kein $t \in T$ kongruent modulo 125.

Sieb mit 7: Die Lösungen von $t^2 \equiv n \equiv 4 \bmod 7$ sind 2 und 5. Es werden also 72, 75, 79, 82 gesiebt.
Aus der Lösung $y = 2$ modulo 7 erhält man die Lösung $z = 23 = 2 + 7 \cdot 3$ modulo 49 ($x = 3$ ist Lösung von $(2 \cdot 2)x \equiv 4837/7 \equiv 5 \bmod 7$). Die andere Lösung ist $26 \equiv -23 \bmod 49$. Es werden also 72 und 75 gesiebt.
Die Lösungen modulo 343 sind 72 und 271. Gesiebt wird 72.
Die Lösungen modulo 2401 sind 614 und 1787. Hierzu ist kein $t \in T$ kongruent modulo 2401.

Sieb mit 11: Die Lösungen von $t^2 \equiv n \equiv 1 \bmod 11$ sind 1 und 10. Es werden also 76, 78 gesiebt.
Die Lösungen modulo 121 sind 1 und 120. Hierzu ist kein $t \in T$ kongruent modulo 121.

Sieb mit 17: Die Lösungen von $t^2 \equiv n \equiv 13 \bmod 17$ sind 8 und 9. Es werden also 76, 77 gesiebt.
Die Lösungen modulo 289 sind 144 und 145. Hierzu ist kein $t \in T$ kongruent modulo 289.

Sieb mit 31: Die Lösungen von $t^2 \equiv n \equiv 5 \bmod 31$ sind 6 und 25. Hierzu ist kein $t \in T$ kongruent modulo 31.

Sieb mit 43: Die Lösungen von $t^2 \equiv n \equiv 25 \bmod 43$ sind 5 und 38. Es wird also 81 gesiebt.
Die Lösungen von $t^2 \equiv n \equiv 25 \bmod 43$ sind 253 und 1596. Hierzu ist kein $t \in T$ kongruent modulo 43^2.

Sieb mit 2: Es gilt $n \equiv 1 \bmod 8$. Falls $s(t)$ gerade ist, so ist der Faktor 2 mindestens dreifach enthalten. Die Lösungen von $t^2 \equiv n \equiv 1 \bmod 8$ sind $1, 3, 5, 7$. Gesiebt werden also $71, 73, 75, 77, 79, 81, 83, 85$.
Die Lösungen von $t^2 \equiv n \equiv 9 \bmod 16$ sind $3, 5, 11, 13$. Gesiebt werden also $75, 77, 83, 85$.
Die Lösungen von $t^2 \equiv n \equiv 9 \bmod 32$ sind $3, 13, 19, 29$. Gesiebt werden also $77, 83$.
Die Lösungen von $t^2 \equiv n \equiv 41 \bmod 64$ sind $13, 19, 45, 51$. Gesiebt werden also $77, 83$.
Die Lösungen von $t^2 \equiv n \equiv 105 \bmod 128$ sind $19, 45, 83, 109$. Gesiebt wird nur 83. Man stellt fest, daß $s(83)$ den Faktor 2^{11} enthält.

Man erhält die folgende Matrix M.

	$s(t)$	2	5	7	11	17	31	43	$s'(t)$
70	59								59
71	200	3	2						1
72	343			3					1
73	488	3							61
74	635		1						127
75	784	4		2					1
76	935		1		1	1			1
77	1088	6				1			1
78	1243				1				113
79	1400	3	2	1					1
80	1559								1559
81	1720	3	1					1	1
82	1883			1					269
83	2048	11							1
84	2215		1						443
85	2384	4							149

Die Menge BZ ist somit $\{71, 72, 75, 76, 77, 79, 81, 83\}$. Die zugehörigen Vektoren ϵ_i sind

$$(1,0,0,0,0,0,0),\ (0,0,1,0,0,0,0),\ (0,0,0,0,0,0,0),\ (0,1,0,1,1,0,0),$$
$$(0,0,0,0,1,0,0),\ (1,0,1,0,0,0,0),\ (1,1,0,0,0,0,1),\ (1,0,0,0,0,0,0).$$

Darunter sind einige Teilmengen linear abhängig, z. B. die Vektoren zu 71 und 83. Hiermit erhält man $(71 \cdot 83)^2 \equiv (2^7 \cdot 5)^2 \bmod n$ und als Teiler $\operatorname{ggt}(1052 + 640, 4841) = 47$. Ferner ist das zu 75 gehörige ϵ schon der Nullvektor. Dies liefert $75^2 \equiv 28^2 \bmod n$ und den Teiler $\operatorname{ggt}(75 + 28, 4841) = 103$. Die Zerlegung von 4841 ist somit $47 \cdot 103$.

8 Verfahren zur Berechnung von Diskreten Logarithmen

Hier sollen Verfahren besprochen werden, die diskrete Logarithmen in zyklischen endlichen abelschen Gruppen bestimmen. Die Schwierigkeit, diskrete Logarithmen berechnen zu können, ist die Grundlage für die Sicherheit des ElGamal Verfahrens und vieler anderer kryptographischer Verfahren. Im folgenden sei G immer eine zyklische abelsche Gruppe der Ordnung n mit einem Erzeuger g. Das Problem des Diskreten Logarithmus besteht darin, für ein $A \in G$ ein $a \in \{0, 1, \ldots, n-1\}$ zu finden mit

$$g^a = A.$$

Die einfachste Methode, a zu bestimmen, besteht darin, für alle $x = 0, 1, 2, \ldots$ das g^x zu berechnen und zu prüfen, ob es mit A übereinstimmt. Dies benötigt i. a. aber $O(n)$ Multiplikationen. D. h. schon für relativ kleine Gruppen ist dies in der Praxis nicht durchführbar.

Beispiel: $G = (\mathbb{Z}/1697\backslash\{0\}, \cdot)$ mit dem Erzeuger $g = 3$. Es soll der Logarithmus von $A = 8$, $\log_3(8)$, bestimmt werden. Durch Probieren erhält man $a = 1110$. Man benötigt also fast n Multiplikationen in G.

8.1 Babystep–Giantstep–Algorithmus

Eine erste Methode zur schnelleren Berechnung diskreter Logarithmen ist der Babystep–Giantstep Algorithmus von Shank. Man setzt zunächst

$$m = [\sqrt{n}] + 1$$

und macht den Ansatz (Division mit Rest)

$$a = j \cdot m + i, \quad 0 \leq i < m.$$

Es gilt $g^a = A$ genau dann, wenn $g^{jm+i} = A$. Gesucht ist also ein Paar (i, j) mit $g^{mj} = A \cdot g^{-i}$. Dazu berechnet man zunächst die Menge der **Babysteps**

$$B = \{(A \cdot g^{-i}, i) \,:\, i \in \{0, \ldots, m-1\}\}$$

und speichert all diese Paare ab. Dann prüft man für $j = 0, 1, \ldots, m-1$ nach, ob g^{mj} als erste Komponente in einem Element von B vorkommt. Ist dies der Fall, so gilt $g^{jm+i} = A$ und das Problem ist gelöst mit $a = jm + i$.

Algorithmus 8.1 (babystepgiantstep)

Eingabe: Eine zyklische Gruppe G der Ordnung n mit Erzeuger g und ein $A \in G$.
Ausgabe: Ein $a \in \mathbb{N}$ mit $g^a = A$.

begin
$m := [\sqrt{n}] + 1$;
$B := \emptyset$;
inv $:= g^{-1}$;
for $i := 0$ **to** $m - 1$ **do** $B := B \cup (A \cdot \text{inv}^i, i)$;
$M := g^m$;
$b := 1$;
$j := 0$;
while $b \notin B_1 = \{x : \exists i \text{ mit } (x, i) \in B\}$ **do**
begin
$j := j + 1$;
$b := b \cdot M$;
end;
(* Hier ist $M^j = A \cdot g^{-i}$ *)
$a := m \cdot j + i$;
end

Satz 8.2 *Algorithmus 8.1 benötigt $O(\sqrt{n})$ Multiplikationen und $O(\sqrt{n})$ Überprüfungen, ob ein Element als erste Komponente eines Babysteps vorkommt. Ferner müssen $O(\sqrt{n})$ Gruppenelemente gespeichert werden.*

Bemerkung: (1) Algorithmus 8.1 ist wesentlich schneller als die einfache Enumeration.
(2) Die Tests, ob ein b in B_1 liegt, können (nach Sortieren der Elemente von B_1) sehr schnell durchgeführt werden, so daß diese nicht ins Gewicht fallen.
(3) Der Nachteil des Verfahrens ist der sehr hohe Speicheraufwand.

Beispiel: $G = (\mathbb{Z}/1697\backslash\{0\}, \cdot)$ mit dem Erzeuger $g = 3$. Es soll der Logarithmus von $A = 8$, $\log_3(8)$, bestimmt werden. Es gilt $m = 42$ und $g^{-1} = 566$. Die Menge der Babysteps ergibt sich zu

$$B = \{(8,0), (1134,1), (378,2), (126,3), (42,4), (14,5), (1136,6), (1510,7), (1069,8), (922,9), (873,10), (291,11), (97,12), (598,13), (765,14), (255,15), (85,16), (594,17), (198,18), (66,19), (22,20), (573,21), (191,22), (1195,23), (964,24), (887,25), (1427,26), (1607,27), (1667,28), (1687,29), (1128,30), (376,31), (691,32), (796,33), (831,34), (277,35), (658,36), (785,37), (1393,38), (1030,39), (909,40), (303,41)\}.$$

Man berechnet $M = 3^{42} = 1025$ und nacheinander $1025^2 = 182$, $1025^3 = 1577, \ldots$ und prüft, ob diese als erste Komponente eines Babysteps vorkommen. Dies ist das erste Mal der Fall für $1025^{26} = 198$ mit $i = 18$. Also gilt $a = 26 \cdot 42 + 18 = 1110$.

8.2 Der Pollard–ϱ–Algorithmus

Hier soll ein Verfahren angegeben werden, das genauso schnell ist wie der Babystep–Giantstep Algorithmus, das aber nur konstant viel Speicherplatz benötigt. Hierzu wird die Gruppe G zerlegt in drei paarweise disjunkte Mengen G_1, G_2, G_3, d. h. $G_i \cap G_j = \emptyset$, für $i \neq j$ und $G = G_1 \cup G_2 \cup G_3$. Ferner sei die Funktion $f : G \to G$ definiert durch

$$f(x) := \begin{cases} g \cdot x, & \text{falls } x \in G_1 \\ x^2, & \text{falls } x \in G_2 \\ A \cdot x, & \text{falls } x \in G_3 \end{cases} .$$

Wir wählen eine Zahl $u_0 \in \{1, \ldots, n\}$ zufällig und setzen $x_0 := g^{u_0}$. Die Folge $\{x_i\}$ wird dann rekursiv berechnet durch

$$x_{i+1} = f(x_i), \quad i = 0, 1, \ldots$$

Alle Folgenglieder lassen sich darstellen als

$$x_i = g^{u_i} \cdot A^{v_i}$$

(mit $v_0 = 0$). Die u_i und v_i lassen sich rekursiv bestimmen durch

$$u_{i+1} = \begin{cases} u_i + 1 \bmod n, & \text{falls } x_i \in G_1 \\ 2u_i \bmod n, & \text{falls } x_i \in G_2 \\ u_i, & \text{falls } x_i \in G_3 \end{cases} \quad \text{und} \quad v_{i+1} = \begin{cases} v_i, & \text{falls } x_i \in G_1 \\ 2v_i \bmod n, & \text{falls } x_i \in G_2 \\ v_i + 1 \bmod n, & \text{falls } x_i \in G_3 \end{cases} . \tag{8.1}$$

Da G endlich ist, gilt irgendwann $x_{j+k} = x_j$, für ein $j \geq 0$ und $k \geq 1$.
Um aber nicht alle bisherigen x_i abspeichern zu müssen, benutzt man den gleichen Trick wie bei dem Pollardschen ϱ–Algorithmus zur Faktorisierung. Man berechnet nämlich parallel die Folgen x_i und $y_i = x_{2i}$ und wie in Satz 7.3 ist gewährleistet, daß im Falle $x_j = x_{j+k}$ auch bald $x_i = x_{2i}$ für ein i gilt. In diesem Falle folgt dann

$$g^{u_i} A^{v_i} = g^{u_{2i}} A^{v_{2i}}, \quad \text{also} \quad g^{u_i - u_{2i}} = A^{v_{2i} - v_i} = g^{a(v_{2i} - v_i)}.$$

Der diskrete Logarithmus a von A (zur Basis g) ist also Lösung der Kongruenz

$$(v_{2i} - v_i) \cdot a \equiv (u_i - u_{2i}) \bmod n.$$

Ist diese Kongruenz nicht eindeutig lösbar, so muß man die richtige Lösung durch Ausprobieren ermitteln. Geht dies nicht effizient genug, weil es zu viele Lösungen gibt, so sollte man mit einem neuen Startwert beginnen.

Algorithmus 8.3 (pollard)

Eingabe: Eine zyklische Gruppe G der Ordnung n mit Erzeuger g und ein $A \in G$.
Ausgabe: Ein $a \in \mathbb{N}$ mit $g^a = A$.

begin
  Wähle eine Partition (G_1, G_2, G_3) von G;
  Wähle ein $u_0 \in \{1, \ldots, n\}$ zufällig;
  $v_0 := 0$;
  $x_0 := y_0 := g^{u_0}$;
  $i := 0$;
  repeat
    $x_{i+1} := f(x_i)$;
    $y_{i+1} := f(f(y_i))$;
    berechne u_{i+1}, v_{i+1} und $u_{2(i+1)}, v_{2(i+1)}$ mittels (8.1);
    $i := i + 1$;
  until $x_i = y_i$;
  $d :=$ ggt $(v_{2i} - v_i, n)$;
  if $d <$ schranke **then**
  begin
    z sei modulo n/d eindeutige Lösung von $(v_{2i} - v_i)/d \cdot z \equiv (u_i - u_{2i})/d \bmod n/d$;
    while $g^z \neq A$ **do** $z := z + n/d$;
    $a := z$;
  end else 'Neuer Versuch';
end

Bemerkung: Natürlich hängt die Dauer von Algorithmus 8.3 davon ab, wie schnell zwei Gruppenelemente x_i gleich werden. Nimmt man an, daß sich die Funktion $f : G \to G$ wie eine Zufallsfunktion verhält, so kann man auf der Basis von Satz 7.2 zeigen, daß etwa $O(\sqrt{n})$ Folgenglieder berechnet werden müssen. Es werden daher $O(\sqrt{n})$ Gruppenoperationen benötigt, während der Bedarf an Speicherplatz zu vernachlässigen ist.

Beispiel: $G = (\mathbb{Z}/1697\backslash\{0\}, \cdot)$ mit dem Erzeuger $g = 3$. Beachte: $n = 1696$. Es soll der Logarithmus von $A = 8$, $a = \log_3(8)$, bestimmt werden. Wir wählen $G_1 = \{x : x \equiv 1 \bmod 3\}$, $G_2 = \{x : x \equiv 0 \bmod 3\}$, $G_3 = \{x : x \equiv 2 \bmod 3\}$ und $u_0 = 123$. Dann ergibt sich die folgende Tabelle

i	x_i	y_i	u_i	v_i	u_{2i}	v_{2i}
0	1504	1504	123	0	123	0
1	1118	459	124	0	124	1
2	459	759	124	1	249	2
3	253	429	248	2	996	8
4	759	1457	249	2	592	32
5	798	1028	498	4	593	33
6	429	1306	996	8	593	35
7	765	798	296	16	594	36
8	1457	765	592	32	680	144
9	1474	1474	592	33	1360	289

Zunächst gilt $d = \text{ggt}\,(289-33, 1696) = 32$. Die Lösung von $(289-33)/32 \cdot z \equiv (592-1360)/32 \bmod 53$ ist $z = -3 \equiv 50 \bmod 53$. Durch Probieren von $a = 50, 50+53, 50+2\cdot 53, \ldots$, erhält man schließlich $a = 50 + 20 \cdot 53 = 1110$.

8.3 Der Pohlig–Hellman–Algorithmus

Das Verfahren von Pohlig–Hellman beruht darauf, die Berechnung eines Diskreten Logarithmus in einer großen Gruppe auf die Berechnung mehrerer in kleineren Gruppen zurückzuführen. Wir setzen im folgenden voraus, daß die Fakorisierung der Gruppenordnung bekannt ist.

Lemma 8.4 *Sei G eine zyklische Gruppe der Ordnung $n = p_1^{k_1} \cdot \ldots \cdot p_r^{k_r}$ mit Erzeuger g. Für $i = 1, \ldots, r$ sei $q_i = p_i^{k_i}$, $n_i = n/q_i$ und*

$$S_i = \{x^{n_i} : x \in G\}, \quad T_i = \{x \in G : x^{q_i} = 1\}.$$

Dann ist $S_i = T_i$ eine Untergruppe von G der Ordnung q_i, die von $g_i := g^{n_i}$ erzeugt wird.

Satz 8.5 *Seien G, n, g, q_i, n_i, g_i wie im vorigen Lemma und $A \in G$. Ferner seien $a_1, \ldots, a_r \in G$ mit $g_i^{a_i} = A_i := A^{n_i}$. Dann gilt*

$$g^a = A \quad \Leftrightarrow \quad a \equiv a_i \bmod q_i,\ i = 1, \ldots, r.$$

Man kann also das Problem in G reduzieren auf das Problem, die diskreten Logarithmen a_i in den Gruppen T_i mit Primzahlpotenzordnung zu bestimmen. In einem nächsten Schritt wird das Problem auf Gruppen mit Primzahlordnung reduziert. Sei G eine zyklische Gruppe der Ordnung $n = p^k$, g ein Erzeuger von G und $A \in G$. Gesucht ist wieder ein $a \in \mathbb{N}$ mit $g^a = A$.

Lemma 8.6 *Die Menge $T = \{x \in G : x^p = 1\} = \{x^{p^{k-1}} : x \in G\}$ ist eine Untergruppe von G der Ordnung p mit Erzeuger $\gamma = g^{p^{k-1}}$.*

Wir verwenden den Ansatz (p–adische Entwicklung)

$$a = x_0 + x_1 \cdot p + x_2 \cdot p^2 + \ldots + x_{k-1} \cdot p^{k-1}, \quad x_i \in \{0, 1, \ldots, p-1\}, \quad i = 0, \ldots, k-1.$$

Man kann die Koeffizienten x_i nun nacheinander bestimmen. Zunächst folgt aus $g^a = A$ durch Potenzieren mit p^{k-1}, daß

$$\gamma^{x_0} = (g^{p^{k-1}})^{x_0} = g^{a p^{k-1}} = A^{p^{k-1}}.$$

x_0 kann also durch Berechnung eines diskreten Logarithmus in der Untergruppe T bestimmt werden. Angenommen, $x_0, x_1, \ldots, x_{i-1}$ sind schon bestimmt. Dann läßt sich $g^a = A$ schreiben als

$$g^{x_i p^i + \ldots + x_{k-1} p^{k-1}} = A \cdot g^{-(x_0 + x_1 p + \ldots + x_{i-1} p^{i-1})}.$$

Potenzieren mit p^{k-i-1} ergibt (unter Beachtung von $g^n = 1$)

$$\gamma^{x_i} = (g^{p^{k-1}})^{x_i} = (A \cdot g^{-(x_0 + x_1 p + \ldots + x_{i-1} p^{i-1})})^{p^{k-i-1}}.$$

Damit läßt sich x_i durch Berechnung eines diskreten Logarithmus in T bestimmen.

Algorithmus 8.7 (pohlighellman)

Eingabe: Eine zyklische Gruppe G der Ordnung n mit Erzeuger g und ein $A \in G$.
Ausgabe: Ein $a \in \mathbb{N}$ mit $g^a = A$.

begin
bestimme die Primfaktorzerlegung von $n = p_1^{k_1} \cdot \ldots \cdot p_r^{k_r}$;
for $j := 1$ **to** r **do**
begin
$A_j := A^{n/p_j^{k_j}}$;
$g_j := g^{n/p_j^{k_j}}$;
$k := k_j$; $p := p_j$; $A := A_j$; $g := g_j$; (Vereinfachung der Bezeichnung)
$\gamma = g^{p^{k-1}}$;
$x_{-1} := 0$;
$s := 1$;
for $i := 0$ **to** $k-1$ **do**
begin
$s := s \cdot g^{-x_{i-1} p^{i-1}}$;
$B := (A \cdot s)^{p^{k-i-1}}$;
$x_i := \log_\gamma(B)$;
end;
$a_j := x_0 + x_1 \cdot p + \ldots + x_{k-1} p^{k-1}$;
end;
bestimme a mit $a \equiv a_j \bmod p_j^{k_j}$ mit dem chinesischen Restsatz;
end

Bemerkung: Algorithmus 8.7 ist anwendbar, wenn alle auftretenden Primteiler p klein sind. Zur Bestimmung des Diskreten Logarithmus in der kleineren Gruppe kann man andere Verfahren anwenden oder eventuell eine vollständige Tabelle aufstellen.

Beispiel: $G = (\mathbb{Z}/1697\backslash\{0\}, \cdot)$ mit dem Erzeuger $g = 3$. Beachte: $n = 1696 = 53 \cdot 2^5$. Es soll der Logarithmus von $A = 8$, $a = \log_3(8)$, bestimmt werden. Es ist $n_1 = 53, n_2 = 32$ und damit $g_1 = 3^{53} = 69$, $A_1 = 8^{53} = 1328$ und $g_2 = 3^{32} = 1336$, $A_2 = 8^{32} = 577$. Zu lösen sind also

$$69^{a_1} \equiv 1328 \bmod 1697 \quad \text{und } 1336^{a_2} \equiv 577 \bmod 1697 \quad \text{mit } a_1 \in \{0, \ldots, 31\},\ a_2 \in \{0, \ldots, 52\}.$$

Letzteres Problem kann man z. B. mit dem Babystep–Giantstep Algorithmus lösen. Hier ist $n = 52$, $g = 1336$ und $A = 577$, also $g^{-1} = 1147$ und $m = 8$. Die Menge der Babysteps ergibt sich zu

$$B = \{(577, 0), (1686, 1), (959, 2), (317, 3), (441, 4), (121, 5), (1330, 6), (1604, 7), (240, 8)\}.$$

Man berechnet $M = 1336^8 = 561$ und nacheinander $561^2 = 776$, $561^3 = 904$, $561^4 = 1438$, $561^5 = 643$, $561^6 = 959$. Dies ist erste Komponente von $(959, 2)$, also $j = 6$, $i = 2$ und damit $a_2 = 6 \cdot 8 + 2 = 50$.
Nun zur Bestimmung von a_1: Hier ist $n = 32$, $p = 2$, $k = 5$, $g = 69$, $A = 1328$, $g^{-1} = 910$. Man bestimmt $\gamma = g^{16} = -1$. Die Berechnung des Diskreten Logarithmus $\log = \log_\gamma$ in $T = \{\pm 1\}$ ist sehr einfach: $\log(1) = 0$ und $\log(-1) = 1$.

Für $i = 0$ ist $s = 1, B = A^{16} = 1$, also $x_0 = \log 1 = 0$.
Für $i = 1$ ist $s = 1, B = A^8 = -1$, also $x_1 = \log(-1) = 1$.
Für $i = 2$ ist $s = 1 \cdot 910^2 = 1661, B = (As)^4 = (1405)^4 = -1$, also $x_2 = \log(-1) = 1$.
Für $i = 3$ ist $s = 1661 \cdot 910^4 = 860, B = (As)^2 = (-1)^2 = 1$, also $x_3 = \log 1 = 0$.
Für $i = 4$ ist $s = 860 \cdot 910^0 = 860, B = (As)^1 = (-1)^1 = -1$, also $x_4 = \log(-1) = 1$.
Es ergibt sich

$$a_1 = x_0 + 2x_1 + 4x_2 + 8x_3 + 16x_4 = 2 + 4 + 16 = 22.$$

Schließlich erhält man a durch Lösen des Kongruenzensystems $a \equiv 22 \bmod 32,\ a \equiv 50 \bmod 53$ mit dem chinesischen Restsatz. Die Lösung und damit der Diskrete Logarithmus ist $a = 1110$.

8.4 Index–Rechnung

Der Index–Rechnung Algorithmus ist das schnellste bisher bekannte Verfahren zur Bestimmung von diskreten Logarithmen (in $\mathbb{F}_q^*$ auch Index genannt). Es ist verwandt mit den Methoden der Faktorisierung mit Hilfe von Faktorbasen. Sei G zyklische Gruppe der Ordnung n mit Erzeuger g. Man wählt zunächst eine geeignete **Faktorbasis** $B = \{b_1, \ldots, b_r\} \subset G$.

Definition 8.8 *Sei $B = \{b_1, \ldots, b_r\} \subset G$ eine Faktorbasis. Dann heißt $x \in G$* **B–Element**, *wenn es ganze Zahlen $x_1, \ldots, x_r \geq 0$ gibt mit*

$$x = \prod_{i=1}^{r} b_i^{x_i}.$$

Bemerkung: Ist $G = \mathbb{Z}/m$, für eine ganze Zahl m, so spricht man auch von **B–Zahlen**.

In einem ersten Schritt bestimmt man die Logarithmen $l_1, \ldots, l_r$ der Basiselemente $b_1, \ldots, b_r$, d. h. $b_i = g^{l_i}$, für $i = 1, \ldots, r$. Dazu wählt man zufällige Zahlen $k \in \{0, \ldots, n-1\}$ und prüft, ob g^k ein B–Element ist. Im positiven Fall schreibt man

$$g^k = \prod_{i=1}^{r} b_i^{k_i} = \prod_{i=1}^{r} g^{k_i l_i} = g^{\sum_{i=1}^{r} k_i l_i}.$$

Hieraus folgt die Kongruenz

$$k \equiv k_1 l_1 + \ldots + k_r l_r \bmod n.$$

Von diesen sogenannten Relationen sammelt man so viele, daß die $l_1, \ldots, l_r$ eindeutig bestimmt sind, also r Stück, oder besser einige mehr. Dann löst man das lineare Kongruenzensystem nach den $l_1, \ldots, l_r$ auf (z. B. mit dem Gaußalgorithmus).
Man beachte, daß dieser Vorgang nur einmal pro Gruppe durchgeführt werden muß. Dann ist man in der Lage, mit relativ wenig Aufwand den Logarithmus eines beliebigen Elementes $A \in G$ zu bestimmen. Man wählt ein zufälliges $x \in \{0, \ldots, n-1\}$ und berechnet $A \cdot g^x$. Dies macht man solange, bis $A \cdot g^x$ ein B–Element ist. Dann gilt also für geeignete ganze Zahlen $x_1, \ldots, x_r \geq 0$

$$A \cdot g^x = \prod_{i=1}^{r} b_i^{x_i} = \prod_{i=1}^{r} g^{l_i x_i} = g^{\sum_{i=1}^{r} l_i x_i}.$$

Mit $A = g^a$ folgt dann

$$a \equiv \sum_{i=1}^{r} x_i l_i - x \bmod n.$$

Es ergibt sich der folgende allgemeine Algorithmus.

Algorithmus 8.9 (index)

Eingabe: Eine zyklische Gruppe G der Ordnung n mit Erzeuger g und ein $A \in G$.
Ausgabe: Ein $a \in \mathbb{N}$ mit $g^a = A$.

begin
  wähle eine 'geeignete' Faktorbasis $B = \{b_1, \ldots, b_r\} \subset G$;
  (* Suchen nach Relationen *)
  $z := 0$;
  repeat
    wähle $k \in \{0, \ldots, n-1\}$ zufällig;
    berechne g^k;
    if g^k B–Element **then**
    begin
      schreibe $g^k = \prod_{i=1}^{r} b_i^{k_i}$ mit $k_i \geq 0$;
      erhalte die Kongruenz KO_z: $k \equiv \sum_{i=1}^{r} k_i l_i \bmod n$;
      $z := z + 1$;
    end;
  until $z = r + c$; (mit einer kleinen Konstanten c, z. B. $c = 10$)
  (* Bestimmung der Logarithmen der Basis *)
  Löse das Kongruenzensystem $\mathrm{KO}_0, \ldots, \mathrm{KO}_{r+c-1}$ mit den Unbekannten $l_1, \ldots, l_r$;
  (* Bestimmung von a *)
  repeat
    wähle $x \in \{0, \ldots, n-1\}$ zufällig;
  until $A \cdot g^x$ ist B–Element;
  schreibe $A \cdot g^x = \prod_{i=1}^{r} b_i^{x_i}$, mit $x_i \geq 0$;
  $a := \sum_{i=1}^{r} l_i x_i - x \bmod n$;
end

Für die Gruppe G sollten zwei Bedingungen erfüllt sein:

1. Die Faktorbasis sollte effizient wählbar sein.

2. Es sollte sich leicht nachprüfen lassen, ob ein Gruppenelement ein B–Element ist.

Für die Gruppen $\mathbb{F}_p^* = (\mathbb{Z}/p\backslash\{0\}, \cdot)$ und die Einheitengruppe $\mathbb{F}_q^*$ eines endlichen Körpers (mit $q = p^r$) ist beides möglich.
Im ersten Fall wählt man B als die Menge aller Primzahlen unterhalb einer Schranke T. Die Überprüfung, ob ein Element ein B–Element ist, erfolgt durch Probedivision durch all diese Primzahlen.
Im zweiten Fall kann ein Element von G mit einem Polynom aus $\mathbb{F}_p[x]$ mit Grad $< r$ identifiziert werden. Als Faktorbasis nimmt man alle irreduziblen Polynome vom Grad kleiner einer

Schranke t. Man überprüft, ob ein Element ein B–Element ist, indem man das Polynom zur Probe durch die irreduziblen Polynome aus B (polynom–) dividiert.
Bei einer geschickten Auswahl der Schranken T bzw. t ist Algorithmus 8.9 dann das effizienteste Verfahren zur Bestimmung Diskreter Logarithmen in $\mathbb{F}_p^*$ bzw. $\mathbb{F}_q^*$. Die Anzahl der Basiselemente sollte einerseits nicht zu groß sein, damit die Berechnung der l_i nicht zu lange dauert. Andererseits sollte die Anzahl nicht zu klein sein, damit relativ viele Gruppenelemente B–Elemente sind.

Beispiel: $G = \mathbb{F}_{1697}^*$ (d. h. $n = 1696$), $A = 8$, $g = 3$. Als Faktorbasis wählen wir $B = \{2, 3, 5, 7\}$. Wir erhalten die folgenden vier Relationen zu den Elementen der Faktorbasis (Fehlversuche werden weggelassen):

$$\begin{aligned} 3^{150} \bmod 1697 &= 168 = 2^3 \cdot 3 \cdot 7 \\ 3^{388} \bmod 1697 &= 960 = 2^6 \cdot 3 \cdot 5 \\ 3^{520} \bmod 1697 &= 336 = 2^4 \cdot 3 \cdot 7 \\ 3^{1343} \bmod 1697 &= 80 = 2^4 \cdot 5 \end{aligned}$$

Es ergibt sich das folgende Kongruenzensystem modulo 1696:

$$\begin{aligned} 150 &\equiv 3l_1 + l_2 + l_4 \bmod 1696 \\ 388 &\equiv 6l_1 + l_2 + l_3 \bmod 1696 \\ 520 &\equiv 4l_1 + l_2 + l_4 \bmod 1696 \\ 1343 &\equiv 4l_1 + l_3 \bmod 1696 \end{aligned}$$

Die Lösung dieses Sytems ist

$$l_1 = 370,\ l_2 = 1,\ l_3 = 1559,\ \text{und}\ l_4 = 735.$$

Sei $x = 821$ gewählt. Dann folgt aus

$$Ag^x = 90 = 2 \cdot 3^2 \cdot 5$$

mit $x_1 = 1,\ x_2 = 2,\ x_3 = 1,\ x_4 = 0$, daß

$$a = (l_1 + 2l_2 + l_3 - 821) \bmod 1696,$$

also $a = 1110$.

Beispiel: $G = \mathbb{F}_{2^7}^*$, also $n = 127$. $f(x) = x^7 + x + 1$ ist ein irreduzibles Polynom über $\mathbb{F}_2$. Ein Erzeuger g ist das Polynom x. Es soll der Logarithmus von $A = x^4 + x^3 + x^2 + x + 1$ zur Basis x bestimmt werden. Als Faktorbasis wählen wir alle irreduziblen Polynome vom Grad ≤ 3:

$$B = \{x, x+1, x^2+x+1, x^3+x+1, x^3+x^2+1\}.$$

Wir erhalten die folgenden fünf Relationen zu den Elementen der Faktorbasis (Fehlversuche werden weggelassen):

$$\begin{aligned} x^{18} \bmod f(x) &= x^6 + x^4 &&= x^4 \cdot (x+1)^2 \\ x^{105} \bmod f(x) &= x^6 + x^5 + x^4 + x &&= x \cdot (x+1)^2 \cdot (x^3 + x^2 + 1) \\ x^{72} \bmod f(x) &= x^6 + x^5 + x^3 + x^2 &&= x^2 \cdot (x+1)^2 \cdot (x^2 + x + 1) \\ x^{45} \bmod f(x) &= x^5 + x^2 + x + 1 &&= (x+1)^2 \cdot (x^3 + x + 1) \\ x^{121} \bmod f(x) &= x^6 + x^5 + x^4 + x^3 + x^2 + x + 1 &&= (x^3 + x + 1) \cdot (x^3 + x^2 + 1) \end{aligned}$$

Es ergibt sich das folgende Kongruenzensystem modulo 127:

$$\begin{aligned} 18 &\equiv 4l_1 + 2l_2 && \bmod 127 \\ 105 &\equiv l_1 + 2l_2 + l_5 && \bmod 127 \\ 72 &\equiv 2l_1 + 2l_2 + l_3 && \bmod 127 \\ 45 &\equiv 2l_2 + l_4 && \bmod 127 \\ 121 &\equiv l_4 + l_5 && \bmod 127 \end{aligned}$$

Die Lösung dieses Systems ist

$$l_1 = 1,\ l_2 = 7,\ l_3 = 56,\ l_4 = 31 \text{ und } l_5 = 90.$$

Sei $x = 66$ gewählt. Dann folgt aus

$$Ag^x = (x^4 + x^3 + x^2 + x + 1) \cdot x^{66} \bmod f(x) = x^5 + x^3 + x = x \cdot (x^2 + x + 1)^2,$$

mit $x_1 = 1,\ x_2 = 0,\ x_3 = 2, x_4 = x_5 = 0$, daß

$$a = (l_1 + 2l_3 - 66) \bmod 127,$$

also $a = 47$.

9 Elliptische Kurven

9.1 Grundlegende Tatsachen

Im folgenden sei K ein Körper ($\mathbb{R}$, $\mathbb{C}$, $\mathbb{Q}$ oder $\mathbb{F}_q$ mit $q = p^r$ und $p \notin \{2, 3\}$). Hier werden zunächst wesentliche Ergebnisse (zum Teil ohne Beweis) skizziert.

Definition 9.1 *Sei K ein Körper (der Charakteristik $\neq 2, 3$). Ferner sei $x^3 + ax + b$ (mit $a, b \in K$) ein kubisches Polynom ohne mehrfache Nullstellen. Dann heißt*

$$E = \{(x, y) \in K^2 \,:\, y^2 = x^3 + ax + b\} \tag{9.1}$$

zusammen mit einem ausgezeichneten Punkt $\mathcal{O}$ (im Unendlichen) **elliptische Kurve** *über K.*

Bemerkung: Die allgemeine Form einer elliptischen Kurve ist

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

Diese kann im Fall, daß die Charakteristik $\neq 2, 3$ ist, auf die Form (9.1) gebracht werden. Alle Ergebnisse lassen sich auch für Körper mit Charakteristik 2 oder 3 formulieren und beweisen.

Definition 9.2 *Sei $F(x, y) = y^2 - x^3 - ax - b$. Ein Punkt $(x, y) \in E$ heißt* **nichtsingulär**, *falls* $\operatorname{grad} F(x, y) \neq (0, 0)$.

Der Gradient besteht hier aus den formalen partiellen Ableitungen und kann bezüglich jedem Körper gebildet werden.

Lemma 9.3 *Die folgenden Aussagen sind äquivalent:*

(a) $x^3 + ax + b$ *hat keine doppelten Nullstellen.*

(b) Alle Punkte in E sind nichtsingulär.

(c) $4a^3 + 27b^2 \neq 0$.

Die wichtigste Eigenschaft von elliptischen Kurven ist, daß sie abelsche Gruppe bilden. Die Gruppenoperation wird gegeben durch die folgende Definition.

Definition 9.4 *Es seien $P = (x_1, y_1), Q = (x_2, y_2)$ zwei Punkte der elliptischen Kurve E.*

(a) Falls $P = \mathcal{O}$ (oder $Q = \mathcal{O}$), dann sei $P + Q = Q$ (bzw. $P + Q = P$).

(b) Falls $(x_2, y_2) = (x_1, -y_1)$, dann setze $P + Q = \mathcal{O}$.

(c) Ansonsten sei $R = P + Q = (x_3, y_3)$ *definiert durch*

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & \text{falls } P \neq Q, \\ \frac{3x_1^2 + a}{2y_1}, & \text{falls } P = Q \end{cases} \tag{9.2}$$

und

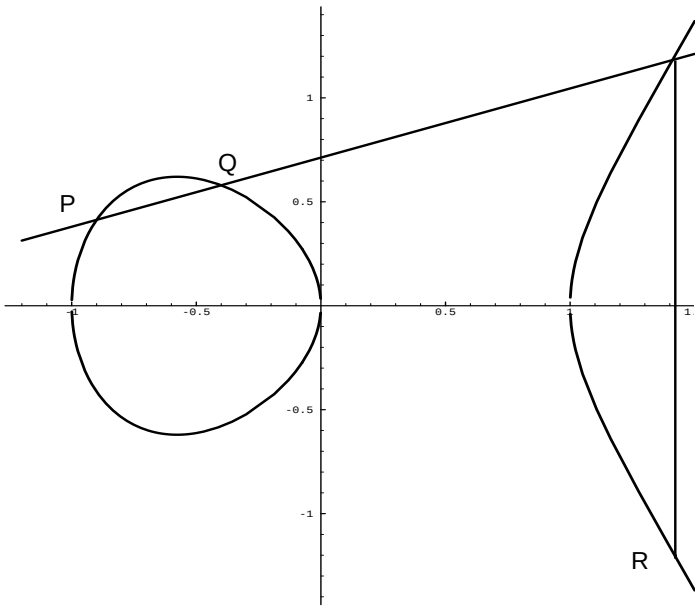
$$x_3 = \lambda^2 - x_1 - x_2, \qquad y_3 = \lambda(x_1 - x_3) - y_1. \tag{9.3}$$

Bemerkung: Die (scheinbar willkürliche) Definition der Addition kann im Fall $K = \mathbb{R}$ durch eine geometrische Konstruktion illustriert werden. Die Gerade durch P und Q schneidet E in genau einem weiteren Punkt. R ist dann die Spiegelung dieses Punktes an der x–Achse. Im Fall $P = Q$ geht die Sekante in eine Tangente in P über.

Satz 9.5 *Es seien* $P = (x_1, y_1)$, $Q = (x_2, y_2) \in E$ *mit* $(x_2, y_2) \neq (x_1, -y_1)$ *und* $g = \{(x, y) \in K^2 : y = \lambda(x - x_1) + y_1\}$. *Dann enthält* $g \cap E$ *außer* P *und* Q *genau einen weiteren Punkt und* $R = P + Q$ *ist die Spiegelung dieses Punktes an der* x*–Achse.*

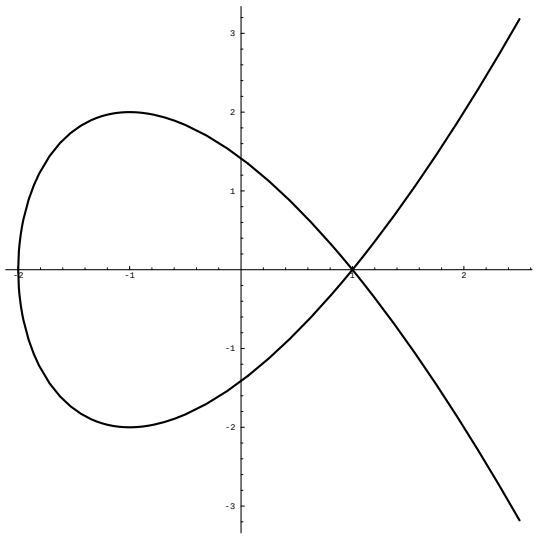
Bemerkung: Im Fall $(x_2, y_2) = (x_1, -y_1)$ ist g eine senkrechte Gerade, die E 'im Unendlichen' in $\mathcal{O}$ schneidet.

Beispiel: Die elliptische Kurve E über $\mathbb{R}$ sei gegeben durch $y^2 = x^3 - x$. Es sollen die beiden Punkte $P = (-0.9, \sqrt{0.171})$ und $Q = (-0.4, \sqrt{0.336})$ addiert werden. Es gilt $\lambda = (\sqrt{0.336} - \sqrt{0.171})/0.5 = 0.332..$ und die Gerade durch P und Q ist gegeben durch $y = 0.332x + 0.71256...$ Als Schnittpunkt ergibt sich $(1.41, 1.394)$, also $R = (1.41, -1.394)$.



Bemerkung: (1) Man kann zeigen, daß eine elliptische Kurve mit der Addition wie in Definition 9.4 eine abelsche Gruppe bildet. Die einzige Schwierigkeit hierbei ist das Assoziativgesetz. Hierfür gibt es außer dem (mühsamen) Beweis durch Nachrechnen noch verschiedene elegantere Beweise, etwa mit Hilfe von doppelperiodischen Funktionen in der Funktionentheorie oder einem Argument aus der Projektiven Geometrie. Das neutrale Element ist per Definition $\mathcal{O}$ und für $P = (x, y)$ ist $-P = (x, -y)$.

(2) Für die Gruppeneigenschaft ist die Bedingung $4a^3 + 27b^2 \neq 0$ notwendig. Z. B. gilt für die Kurve $y^2 = x^3 - 3x + 2 = (x-1)^2 \cdot (x+2)$ über $\mathbb{R}$ (oder auch $\mathbb{F}_p$) mit singulärem Punkt $(1,0)$, daß $(-2,0) + (1,0) = (1,0)$ (die Gerade $y = 0$ enthält nur 2 Punkte, $(1,0)$ wird doppelt gezählt). Dann müßte aber $(-2,0)$ neutrales Element sein. Widerspruch.



Elliptische Kurven in projektiver Form

Der Punkt $\mathcal{O}$ ist der dritte Schnittpunkt der elliptischen Kurve mit einer Senkrechten. Ein natürlicherer Weg, den Punkt $\mathcal{O}$ einzuführen, ist der folgende. Die **Projektive Ebene** ist die Menge aller Äquivalenzklassen von Tripeln $(x,y,z) \neq (0,0,0)$, wobei zwei Tripel als äquivalent gelten, wenn sie kollinear sind, d. h. $(x,y,z) \sim (\lambda x, \lambda y, \lambda z)$. Eine solche Äquivalenzklasse heißt **projektiver Punkt** und entspricht einer Geraden. Die Geraden mit $z \neq 0$ entsprechen genau den Punkten in der Ebene $z = 1$. Die Projektive Ebene besteht also aus dieser 'normalen' affinen Ebene zusammen mit den Geraden mit $z = 0$, die die 'unendlich ferne Gerade' genannt wird. Man kann diese auch als 'Horizont' der Ebene interpretieren. Eine elliptische Kurve kann in projektiver Form $y^2z = x^3 + axz^2 + bz^3$ geschrieben werden. In der Ebene $z = 1$ erhält man die ursprüngliche Gleichung. Für $z = 0$ muß gelten $x^3 = 0$, also $x = 0$. Es kommt also noch zusätzlich die Äquivalenzklasse von $(0,1,0)$ hinzu, die man dann $\mathcal{O}$ nennt.

Elliptische Kurven über $\mathbb{C}$

Hier beschreiben elliptische Kurven 2–dimensionale Flächen im 4–dimensionalen Raum.
Ein **Gitter** L in der komplexen Ebene ist die Menge aller ganzzahligen Linearkombinationen $\alpha w_1 + \beta w_2$ von zwei nichtkollinearen komplexen Zahlen w_1, w_2. Z. B. für $w_1 = 1$ und $w_2 = i$ stellt L die Gaußschen Zahlen dar.
Man kann zeigen, daß es zu einer gegebenen elliptischen Kurve über $\mathbb{C}$ ein Gitter L und eine komplexe Funktion, die sogenannte **Weierstrasssche** $\wp$**–Funktion** (Bez.: $\wp_L(z)$), gibt, mit folgenden Eigenschaften

(1) $\wp(z)$ ist analytisch bis auf Doppelpole in jedem $z \in L$.

(2) $\wp$ genügt der Differentialgleichung $(\wp')^2 = \wp^3 + a\wp + b$, d. h. für jedes $z \notin L$ ist $(\wp(z), \wp'(z))$ in E enthalten.

(3) $z_1, z_2 \in \mathbb{C}$ ergeben genau dann das gleiche $(\wp(z), \wp'(z))$, wenn $z_1 - z_2 \in L$.

(4) Die Abbildung $z \mapsto (\wp(z), \wp'(z))$, für $z \notin L$ und $z \mapsto \mathcal{O}$, für $z \in L$ ergibt eine bijektive Abbildung zwischen E und $\mathbb{C}/L$.

(5) Die Abbildung in (4) ist ein Isomorphismus zwischen abelschen Gruppen (wobei die Addition in $\mathbb{C}/L$ die gewöhnliche Addition in $\mathbb{C}$ modulo L ist).

Eine elliptische Kurve über $\mathbb{C}$ kann also mit einem **Torus** identifiziert werden.

Elliptische Kurven über $\mathbb{F}_q$

Sei $q = p^r$ und p eine Primzahl $\neq 2, 3$. Offensichtlich gibt es in E höchstens $2q + 1$ Elemente, nämlich $\mathcal{O}$ und zu jedem $x \in \mathbb{F}_q$ höchstens zwei y mit $y^2 = x^3 + ax + b$. Ist χ der **quadratische Charakter** von $\mathbb{F}_q$, d. h. für $x \neq 0$ sei $\chi(x) = \pm 1$, je nachdem ob x ein Quadrat in $\mathbb{F}_q$ ist oder nicht (und $\chi(0) = 0$). Dann gilt für die Anzahl N der Punkte in E (inklusive $\mathcal{O}$)

$$N = 1 + \sum_{x \in \mathbb{F}_q} (1 + \chi(x^3 + ax + b)) = q + 1 + \sum_{x \in \mathbb{F}_q} \chi(x^3 + ax + b). \tag{9.4}$$

Genauere Untersuchungen ergeben (**Satz von Hasse**)

$$|N - (q + 1)| \leq 2\sqrt{q}. \tag{9.5}$$

Beispiel: Sei E die elliptische Kurve $y^2 = x^3 + ax + b$ über $\mathbb{F}_{11} (= \mathbb{Z}/11)$. Für gegebenes $x \in \mathbb{F}_{11}$ berechnet man $z = x^3 + x + 6$ und prüft (etwa mit dem Euler–Kriterium), ob z ein quadratischer Rest ist. Im positiven Fall kann man die Wurzeln von z berechnen (da $11 \equiv 3 \bmod 4$, vgl. Rabin Verfahren):

$$\pm z^{(11+1)/4} \bmod 11, \text{ also } \pm z^3 \bmod 11.$$

Dies ergibt die folgende Tabelle

x	$x^3 + x + 6 \bmod 11$	$\left(\frac{z}{p}\right)$	y
0	6	-1	
1	8	-1	
2	5	1	4, 7
3	3	1	5, 6
4	8	-1	
5	4	1	2, 9
6	8	-1	
7	4	1	2, 9
8	9	1	3, 8
9	7	-1	
10	4	1	2, 9

Es gilt also $N = 13$ und

$$E = \{\mathcal{O}, (2,4), (2,7), (3,5), (3,6), (5,2), (5,9), (7,2), (7,9), (8,3), (8,8), (10,2), (10,9)\}.$$

Ferner ist die Gruppe zyklisch, z. B. mit Erzeuger $G = (2, 7)$. Um $2G$ zu berechnen, bestimmt man zunächst

$$\lambda = (3 \cdot 2^2 + 1) \cdot (2 \cdot 7)^{-1} = 2 \cdot 3^{-1} = 2 \cdot 4 = 8.$$

Es ergibt sich

$$x_3 = 8^2 - 2 - 2 = 5, \text{ und } y_3 = 8(2 - 5) - 7 = 2,$$

also $2G = (5, 2)$. Analog erhält man $3G = (8, 3)$, $4G = (10, 2)$, $5G = (3, 6)$, $6G = (7, 9)$, $7G = (7, 2)$, $8G = (3, 5)$, $9G = (10, 9)$, $10G = (8, 8)$, $11G = (5, 9)$, $12G = (2, 4)$.

Die zu einer elliptischen Kurve gehörige abelsche Gruppe ist nicht notwendigerwise zyklisch. Man kann aber zeigen, daß sie ein Produkt von zwei zyklischen Gruppen ist. Genauer ist E isomorph zu

$$\prod_{p|N} \mathbb{Z}/p^\alpha \times \mathbb{Z}/p^\beta,$$

mit gewissen $\alpha \geq 1$ und $\beta \geq 0$. Unter dem **Typ** einer elliptischen Kurve versteht man die Liste $(\dots, p^\alpha, p^\beta, \dots)_{p|N}$. Dieser Typ ist nicht immer einfach zu bestimmen.

9.2 Kryptoverfahren mit Hilfe von elliptischen Kurven

Da eine elliptische Kurve als additive Gruppe geschrieben wird, entspricht hier die Exponentation dem Bilden von Vielfachen kP, für $P \in E$ und $k = 2, 3, \dots$. Diese kann man entsprechend der schnellen Exponentation berechnen, z. B. gilt $100P = 2 \cdot 2 \cdot (16 + 8 + 1)P = 2(2(P + 2(2(2(P + 2P)))))$.

Satz 9.6 *Es sei eine elliptische Kurve E über einem endlichen Körper $\mathbb{F}_q$ gegeben. Zu $P \in E$ kann man kP in einer Zeit von $O(\log k \cdot \log^3 q)$ bestimmen.*

Bemerkung: (1) Kennt man die Anzahl N der Elemente in E, so kann man im Fall $k > N$ den Exponenten wegen $NP = \mathcal{O}$ modulo N reduzieren. Man kann die Laufzeit dann abschätzen durch $O(\log^4 q)$.
(2) Es gibt ein Verfahren von Schoof, das N in einer Zeit von $O(\log^8 q)$ berechnet.

Analogon zum Diffie–Hellman Schlüsselaustausch

Mit der Gruppe E einer elliptischen Kurve über $\mathbb{F}_q$ kann man sofort den Diffie–Hellman Schlüsselaustausch verwirklichen. Da E i. A. nicht zyklisch ist, kann man nicht davon ausgehen, einen Erzeuger zu finden. Es reicht aber, einen Punkt G auf E zu nehmen, der eine 'große' Ordnung hat (am besten in der Größenordnung $N = |E|$). Die Sicherheit beruht dann auf der Schwierigkeit, in der von G erzeugten zyklischen Untergruppe diskrete Logarithmen zu berechnen.

Analogon zum ElGamal Verfahren

Zunächst wählt man einen großen endlichen Körper $\mathbb{F}_q$, eine elliptische Kurve E über $\mathbb{F}_q$ und einen Punkt $G \in E$. Man kann $N = |E|$ z. B. mit dem Verfahren von Schoof bestimmen. Die Kenntnis von N ist aber nicht zwingend.

(1) Wähle ein zufälliges $a \in \{1, \ldots, N-1\}$ (a ist der geheime Schlüssel zum Entschlüsseln) und bestimme $A = aG \in E$ (A ist der öffentliche Schlüssel).

(2) Zu einem zufällig gewählten $k \in \{0, \ldots, N-1\}$ sei $C_{q,E,G,A,k} : E \to E \times E$ definiert durch

$$C_{q,E,G,A,k}(x) = (kG, x + kA).$$

(3) Sei $D_{q,E,a} : E \times E \to E$ definiert durch

$$D_{E,q,a}(y_1, y_2) = y_2 - ay_1.$$

Lemma 9.7 *$D_{q,E,a}(C_{q,E,G,A,k}(x)) = x$, für alle $x \in E$ und alle $k \in \mathbb{N}$.*

Formal ist $\mathcal{P} = E$, $\mathcal{C} = E \times E$,

$$\mathcal{K} = \{(q, E, G, a, A) : G \in E,\ aG = A\}.$$

Für $K = (q, E, G, a, A)$ und $k \in \mathbb{N}$ ist

$$E_K(x) = (kG, x + kA) \quad \text{und} \quad D_K(y_1, y_2) = y_2 - ay_1.$$

Beispiel: $q = p = 11$ und E sei gegeben durch $y^2 = x^3 + x + 6$. Ferner sei $G = (2, 7)$, $a = 7$, also $A = 7G = (7, 2)$. Wählt man $k = 3$, so ist der zum Klartext $x = (10, 9)$ gehörige Schlüsseltext $y = (y_1, y_2)$ mit

$$y_1 = 3(2, 7) = (8, 3) \quad \textit{und} \quad y_2 = (10, 9) + 3(7, 2) = (10, 9) + (3, 5) = (10, 2).$$

Zum Entschlüsseln berechnet man dann

$$x = (10, 2) - 7(8, 3) = (10, 2) - (3, 5) = (10, 2) + (3, 6) = (10, 9).$$

Einbetten von Klartexten

Im obigen Verfahren wird mit Punkten auf einer elliptischen Kurve E gearbeitet. Es muß also zunächst einem Klartext (einer ganzen Zahl $m \in \{0, \ldots, M-1\}$) ein Punkt $P_m \in E$ zugeordnet werden. Dies soll möglichst einfach geschehen und es muß m aus P_m rekonstruierbar sein. Hierfür gibt es keinen deterministischen polynomialen Algorithmus (in $\log q$). Wir werden hier ein probabilistisches Verfahren angeben, dessen Fehlerwahrscheinlichkeit beliebig klein gemacht werden kann.
Man wählt eine ganze Zahl k (die Fehlerwahrscheinlichkeit wird dann 2^{-k}, in der Praxis sollte $k = 30$ oder $k = 50$ ausreichen). Der Klartext bestehe aus ganzen Zahlen $m \in \{0, \ldots, M-1\}$ und q sei so gewählt, daß $q > Mk$.

Für $j = 1, \ldots, k$ schreibt man $m \cdot k + j$ in der p–adischen Entwicklung, also (wegen $(m+1)k < q = p^r$)

$$mk + j = \sum_{i=0}^{r-1} a_i p^i.$$

Das Polynom $a_0 + a_1 x + \ldots + a_{r-1} x^{r-1}$ kann man dann als Element x von $\mathbb{F}_q$ auffassen. Dann berechnet man $z = x^3 + ax + b$ und prüft, ob es ein $y \in \mathbb{F}_q$ gibt mit $y^2 = z$ (man kann in $\mathbb{F}_q$ Wurzeln ziehen wie in $\mathbb{Z}/p$). Im positiven Fall setzt man $P_m = (x, y)$. Ansonsten erhöht man j um 1 usw. Da die Wahrscheinlichkeit, daß z ein Quadrat ist, ungefähr $1/2$ beträgt, findet man mit großer Wahscheinlichkeit einen solchen Punkt P_m. Aus $P_m = (x, y)$ kann man dann m wie folgt rekonstruieren. Zunächst bestimmt man umgekehrt wie oben die ganze Zahl $\tilde{x}$, die zu $x \in \mathbb{F}_q$ gehört. Dann erhält man m durch $m = (\tilde{x} - 1)$ DIV k.

Das Verfahren von Menezes–Vanstone

Bei obigem ElGamal–Typ Verfahren gab es nur $|E| = N \sim q$ mögliche Klartexte. Ferner war es ein Problem, den Klartext in die elliptische Kurve einzubetten. Eine Möglichkeit, dies zu umgehen, ist von Menezes und Vanstone gefunden worden. In dieser Variante wird die elliptische Kurve nur zur 'Maskierung' benutzt. Im Klartext und Schlüsseltext können beliebige Paare von Körperelementen auftauchen.

Gegeben sei ein endlicher Körper $\mathbb{F}_q$ und eine elliptische Kurve E über $\mathbb{F}_q$. Ferner gebe es ein Element $G \in E$ mit einer hinreichend großen Ordnung N, d. h. in der erzeugten zyklischen Untergruppe ist das Problem, diskrete Logarithmen zu berechnen, 'praktisch unlösbar'.

(1) Wähle ein zufälliges $a \in \{1, \ldots, N-1\}$ (a ist der geheime Schlüssel zum Entschlüsseln) und bestimme $A = aG \in E$ (A ist der öffentliche Schlüssel).

(2) Zu einem zufällig gewählten $k \in \{0, \ldots, N-1\}$ sei $C_{q,E,G,A,k} : \mathbb{F}_q^* \times \mathbb{F}_q^* \to E \times \mathbb{F}_q^* \times \mathbb{F}_q^*$ definiert durch

$$C_{q,E,G,A,k}(x_1, x_2) = (kG, c_1 x_1, c_2 x_2),$$

wobei $(c_1, c_2) = kA \in E$.

(3) Sei $D_{q,E,a} : E \times \mathbb{F}_q^* \times \mathbb{F}_q^* \to \mathbb{F}_q^* \times \mathbb{F}_q^*$ definiert durch

$$D_{E,q,a}(Y_0, y_1, y_2) = (y_1 c_1^{-1}, y_2 c_2^{-1}),$$

wobei $(c_1, c_2) = aY_0 \in E$.

Lemma 9.8 *$D_{q,E,a}(C_{q,E,G,A,k}(x)) = x$, für alle $x \in \mathbb{F}_q^* \times \mathbb{F}_q^*$ und alle $k \in \mathbb{N}$.*

Formal ist $\mathcal{P} = \mathbb{F}_q^* \times \mathbb{F}_q^*$, $\mathcal{C} = E \times \mathbb{F}_q^* \times \mathbb{F}_q^*$,

$$\mathcal{K} = \{(q, E, G, a, A) : G \in E,\ aG = A\}.$$

Für $K = (q, E, G, a, A)$ und $k \in \mathbb{N}$ ist

$$E_K(x) = C_{q,E,G,A,k}(x) \quad \text{und} \quad D_K(y) = D_{q,E,a}(y).$$

Beispiel: $q = p = 11$ und E sei gegeben durch $y^2 = x^3 + x + 6$. Man beachte: hier gibt es $10 \cdot 10 = 100$ mögliche Klartexte statt der 13 im Originalverfahren. Es sei $G = (2, 7)$, $a = 7$, also $A = 7G = (7, 2)$. Der Klartext sei $x = (9, 1)$. Man beachte, daß $x \notin E$. Zum Verschlüsseln sei $k = 6$ gewählt worden. Mit $kA = 6(7, 2) = (8, 3)$ ergibt sich $c_1 = 8$ und $c_2 = 3$, also

$$E_K(x) = (6(2, 7), 8 \cdot 9, 3 \cdot 1) = ((7, 9), 6, 3).$$

Zum Entschlüsseln berechnet man zunächst $aY_0 = 7(7, 9) = (8, 3)$ und damit $c_1 = 8$ und $c_2 = 3$. Damit ergibt sich der ursprüngliche Klartext

$$x = (6 \cdot 8^{-1}, 3 \cdot 3^{-1}) = (6 \cdot 7, 3 \cdot 4) = (9, 1).$$

Wahl von $(\mathbf{E}, \mathbf{G})$

In den obigen Verfahren wurde eine elliptische Kurve E und ein Punkt $G \in E$ benötigt. Diese kann man wie folgt gleichzeitig bestimmen.

(1) Wähle $x, y, a \in \mathbb{F}_q$ zufällig.

(2) Setze $b := y^2 - (x^3 + ax)$.

(3) Prüfe, ob $4a^3 + 27b^2 = 0$; Falls ja, gehe zurück zu (1).

Anschließend ist dann der Punkt $G = (x, y)$ auf der elliptischen Kurve E gegeben durch $y^2 = x^3 + ax + b$ enthalten.

Zur Sicherheit

Die Sicherheit der obigen Verfahren beruht auf der Schwierigkeit, in der von G erzeugten Untergruppe diskrete Logarithmen zu bestimmen. Der Punkt G sollte also so gewählt sein, daß die Ordnung dieser Untergruppe einen großen Primfaktor enthält (ansonsten greift der Pohlig–Hellman Algorithmus). Besonders geeignet ist eine elliptische Kurve, deren Ordnung N selbst eine Primzahl ist. Dann ist jedes $G \in E \backslash \{\mathcal{O}\}$ ein Erzeuger.
Es ist wahrscheinlich, daß das Problem des Diskreten Logarithmus für elliptische Kurven schwieriger ist, als das für endliche Körper. Bis 1990 gab es kein solches Verfahren, außer denen, die für beliebige Gruppen funktionieren. Dann aber fanden Menezes, Okamoto und Vanstone ein schnelleres Verfahren für spezielle elliptische Kurven, nämlich für Kurven der Form $y^2 = x^3 + ax$, mit $p \equiv 3 \bmod 4$ (supersinguläre Kurven) und Kurven der Form $y^2 = x^3 + b$ mit $p \equiv 2 \bmod 3$ (anomale Kurven). Bei der Auswahl der Kurven sollte man also solche vermeiden.

Man geht heute davon aus, daß eine elliptische Kurve E, die die gleiche Sicherheit wie 1024–Bit RSA–Systeme bietet, etwa $N = 2^{163}$ Punkte hat. Die Kurve sollte weder supersingulär noch anomal sein und N sollte einen Primfaktor $\geq 2^{160}$ enthalten.

Vorteile von EC–Kryptographie

Public–Key–Kryptographie mit elliptischen Kurven ist die wichtigste bis jetzt bekannte Alternative zu RSA–basierten Verfahren. Solche Alternativen sind dringend nötig, da niemand die Sicherheit von RSA garantieren kann.
Ein Vorteil von elliptischen Kurven ist, daß es zu gegebenem q viele verschiedene elliptische Kurven und viele verschiedene N gibt, aus denen man auswählen kann.
Ein weiterer Grund für die Verwendung von EC–Kryptosystemen besteht darin, daß sie Effizienzvorteile gegenüber RSA–Verfahren bieten. Während nämlich RSA–Verfahren modulare Arithmetik mit 1024–Bit–Zahlen verwenden, reichen für EC–Verfahren schon 163–Bit–Zahlen aus. Zwar ist das Rechnen mit elliptischen Kurven aufwendiger als in primen Restklassengruppen. Jedoch wird dies durch die geringere Länge der verwendeten Zahlen kompensiert. Dadurch ist es z. B. möglich, EC–Kryptographie auf Smart–Cards ohne Koprozessor zu implementieren. Solche Smart–Cards sind wesentlich billiger als Chipkarten mit Koprozessor.

9.3 Ein Primzahltest mit Hilfe von elliptischen Kurven

Der Primzahltest mittels elliptischen Kurven, der hier beschrieben werden soll, ist analog zu dem folgenden Test von **Pocklington**.

Satz 9.9 *Es sei $n \in \mathbb{N}$ und es gebe einen Teiler d von $n-1$ mit $d > \sqrt{n} - 1$ und ein $a \in \{1, \ldots, n-1\}$ mit*

(i) $a^{n-1} \equiv 1 \bmod n$,

(ii) $\text{ggt}\,(a^{(n-1)/q} - 1, n) = 1$, *für alle Primteiler q von d.*

Dann ist n eine Primzahl.

Bemerkung: (1) Insbesondere gilt der Satz, wenn $d = q$ selbst eine Primzahl ist.
(2) Der Test ist probabilistisch nur in dem Sinne, daß a zufällig gewählt wird und geprüft wird, ob es die Bedingung (ii) erfüllt (Man kann davon ausgehen, daß (i) erfüllt ist, da n sonst keine Primzahl ist). Im Gegensatz zu den Tests in Kapitel 6 kann man hier definitiv folgern, daß n eine Primzahl ist (nicht nur wahrscheinlich).

Beispiel: $n = 37861$. $n - 1 = 4 \cdot 3 \cdot 5 \cdot 631$. $d = q = 631$ erfüllt die Voraussetzungen. Mit $a = 2$ gilt $a^{n-1} \equiv 1 \bmod n$ und $a^{(n-1)/q} = 2^{60} \equiv 13262 \bmod n$. Also ist n eine Primzahl.

Im folgenden sei n eine Zahl, die probabilistischen Tests gemäß höchstwahrscheinlich eine Primzahl ist. Man kann zeigen (und es ist nicht von vorneherein klar), daß man zu einem Punkt

$P = (x, y)$, der die Kongruenz $y^2 \equiv x^3 + ax + b \bmod n$ erfüllt, genau wie in Definition 9.4 das Vielfache kP bestimmen kann, unabhängig davon, in welcher Reihenfolge man die Punkte addiert, auch wenn n nicht prim ist. Es könnte nur der Fall eintreten, daß bei der Berechnung von λ der Nenner modulo n nicht invertierbar ist. Dann ist aber n auch nicht prim.

Lemma 9.10 *Es sei $n \in \mathbb{N}$ und p ein Primteiler von n. E sei die Menge von Punkten (x, y) mit $y^2 \equiv x^3 + ax + b \bmod n$ und E' sei die enstprechende elliptische Kurve über $\mathbb{F}_p$. Ferner gelte $\operatorname{ggt}(4a^3 + 27b^2, n) = 1$. Zu einem $P = (x, y) \in E$ sei $P' = (x \bmod p, y \bmod p)$.*

(a) Für $P \in E$ gilt $P' \in E'$.

(b) Gilt $P + Q = \mathcal{O}$ in E, dann gilt in E' auch $P' + Q' = \mathcal{O}$.

(c) Für $P, Q \in E$ mit $P + Q$ definiert und $\neq \mathcal{O}$ gilt $(P + Q)' = P' + Q'$.

(d) Ist kP definiert und $\neq \mathcal{O}$, dann gilt $kP' \neq \mathcal{O}$.

Entsprechend kann man das Verfahren von Schoof zur Bestimmung der Anzahl N modulo n durchführen. Falls n prim ist, so gilt garantiert $N = |E|$. Wieder ist es möglich, daß gewisse Elemente nicht invertierbar sind, wenn n nicht prim ist. Da wir davon ausgehen, daß n höchstwahrscheinlich eine Primzahl ist, wird dieser Fall aber i. a. nicht auftreten.

Nun übernimmt E die Rolle von $\mathbb{Z}/n\backslash\{0\}$ und weiter entspricht $N = |E|$ der Zahl $n - 1 = |\mathbb{Z}/n\backslash\{0\}|$ in Satz 9.9.

Satz 9.11 *Es sei $n \in \mathbb{N}$ und E sei die Menge der Punkte (x, y) mit $y^2 \equiv x^3 + ax + b \bmod n$ und es gelte $\operatorname{ggt}(4a^3 + 27b^2, n) = 1$. Ferner sei $N \in \mathbb{N}$ und d ein Teiler von N mit $d > (n^{1/4} + 1)^2$. Gibt es einen Punkt $P \in E$ mit*

(i) $NP = \mathcal{O}$,

(ii) $(N/q)P$ ist definiert und $\neq \mathcal{O}$, für alle Primteiler q von d,

dann ist n eine Primzahl.

Im Gegensatz zu Satz 9.9 hat man nun den Vorteil, daß man die Parameter a, b, N und P so wählen kann, daß die Bedingungen erfüllt sind. Insbesondere sei dann $N = kq$ mit k 'klein' und q $(= d)$ 'wahrscheinlich prim' und (i) und (ii) seien für ein $P \in E$ erfüllt. Dann ist nach Satz 9.11 n garantiert prim, wenn q prim ist.

Algorithmus 9.12 (ECTest)

Eingabe: Ein $n \in \mathbb{N}$, das 'wahrscheinlich' prim ist;
Ausgabe: n garantiert prim, wenn q prim ist oder 'n nicht prim';

begin
  repeat
    repeat
      repeat
        wähle a, x, y zufällig;
        berechne $b = y^2 - x^3 - ax \bmod n$; $(P = (x, y) \in E)$
      until ggt $(4a^3 + 27b^2, n) = 1$;
      berechne $N = |E|$ (falls dies geht, sonst prim := FALSE);
    until $N = kq$ mit k 'klein' und q 'wahrscheinlich prim';
    Berechne NP und kP (falls dies geht, sonst prim := FALSE);
    if $NP \neq \mathcal{O}$ **then** prim := FALSE;
  until $kP \neq \mathcal{O}$;
end

Das so erhaltene $q = q_1$ ist höchstens halb so groß wie n. Auf dieses q_1 kann man nun das gleiche Verfahren anwenden und erhält q_2 und so weiter. Nach $t = O(\log n)$ Anwendungen von Algorithmus 9.12 erhält man eine Zahl q_t, von der man mit Sicherheit sagen kann, ob sie prim ist. Dann läßt sich rückwärts schließen: q_{t-1} ist prim, ..., $q = q_1$ und damit n ist prim.

Beispiel: $n = 4651$. Wähle $a = 4650 \equiv -1 \bmod n$, $x = 89$ und $y = 109$. Es ergibt sich $b = y^2 - x^3 - ax \bmod n = 0$. Man bestimmt $N = |E| = 4652$ (dies gilt auch nach Übung, wenn man voraussetzt, daß n prim ist). Es gilt $N = 4 \cdot 1163$ und $q = 1163$ ist wahrscheinlich prim. Man kann nachrechnen, daß $N \cdot (x, y) = \mathcal{O}$, aber $4 \cdot (x, y) = (1555, 787) \neq \mathcal{O}$. Also ist n prim, wenn 1163 prim ist.
Für $n' = q = 1163$ wähle $a = 1162 \equiv -1 \bmod n$, $x = 26$ und $y = 97$. Es ergibt sich $b = y^2 - x^3 - ax \bmod n = 0$. Man bestimmt $N = |E| = 1164$ (dies gilt auch nach Übung, wenn man voraussetzt, daß 1163 prim ist). Es gilt $N = 4 \cdot 3 \cdot 97$ und $q = 97$ ist prim. Man kann nachrechnen, daß $N \cdot (x, y) = \mathcal{O}$, aber $12 \cdot (x, y) = (132, 1083) \neq \mathcal{O}$.
Da 97 prim ist, ist auch 1163 und damit 4651 prim.

Bemerkung: (1) Obwohl das Verfahren von Schoof theoretisch polynomial ist, hat es in der Praxis eine relativ hohe Laufzeit. Es gibt eine Variante des obigen Tests von Atkin, in dem dieses Problem umgegangen wird, indem nur spezielle Kurven zugelassen werden, für die N besonders leicht zu berechnen ist.
(2) Das Verfahren ist nicht nachweisbar polynomial. Hierzu bräuchte man eine theoretische Aussage über die Verteilung von 'Fast–Primzahlen' (d. h. $N = kq$, mit k und q wie oben) in dem Intervall $[n + 1 - 2\sqrt{n},\ n + 1 + 2\sqrt{n}]$, die garantiert, daß man schnell eine Kurve finden kann, für die $N = |E|$ fast prim ist. Dies würde aber aus einer sehr plausiblen Vermutung folgen.
(3) Es gibt einen nachweisbar polynomialen probabilistischen Primzahltest von Adleman und Huang. Dieser beruht auf Abelschen Varietäten, einer 2–dimensionalen Verallgemeinerung von elliptischen Kurven. Das Verfahren ist jedoch sehr kompliziert und außerdem für die Praxis nicht brauchbar.

9.4 Faktorisierung mit Hilfe von elliptischen Kurven

Das Verfahren von Lenstra zur Faktorisierung ganzer Zahlen mit Hilfe von elliptischen Kurven beruht auf der gleichen Idee wie das $(p-1)$–Verfahren.
Hier hat man k gleich dem Produkt aller Primzahlpotenzen unterhalb einer Schranke B gesetzt. Hat n einen Primteiler p, so daß $(p-1)$ nur kleine Primfaktoren enthält, so ist $a^k \equiv 1 \bmod p$ und falls $a^k \not\equiv 1 \bmod n$, so ist $\operatorname{ggt}(a^k - 1, n)$ ein echter Teiler von n.
Das $(p-1)$–Verfahren funktioniert, wenn für eine der Gruppen $G(n) = \mathbb{F}_p^* = (\mathbb{Z}/p)\backslash\{0\}$, mit $p|n$, die Ordnung diese Gestalt hat. Diese Gruppen sind aber eindeutig bestimmt durch n und in den meisten Fällen hat keine der Gruppenordnungen die gewünschte Form.
Im Verfahren von Lenstra werden diese Betrachtungen in der Gruppe E einer elliptischen Kurve über einem Primkörper $\mathbb{F}_p$ angestellt. Hier kann man die Gruppenordnung $N = |E|$ durch Auswahl der Parameter a und b (in gewissen Grenzen) variieren und somit erhöht sich die Chance, daß ein solches N nur kleine Primfaktoren enthält.

Gegeben sei eine zusammengesetzte Zahl $n \in \mathbb{N}$. Ferner sei E die Menge von Punkten (x, y) mit $y^2 \equiv x^3 + ax + b \bmod n$, wobei $a, b \in \mathbb{Z}$ und $\operatorname{ggt}(4a^3 + 27b^2, n) = 1$. Es sei p ein Primteiler von n und E' die zu E gehörige elliptische Kurve über $\mathbb{F}_p$ und es sei $N' = |E'|$.
Ist $P = (x, y) \in E$, so gilt für $P' \in E'$ nach dem Satz von Lagrange $N'P' = \mathcal{O}$. Nach Lemma 9.10 (d) ist dann $N'P$ entweder $\mathcal{O}$ oder nicht definiert. Der erste Fall tritt äußerst selten ein (es müßte hier $N'P' = \mathcal{O}$ für alle Primfaktoren gleichzeitig sein). Im zweiten Fall heißt das aber, daß bei der Berechnung von $N'P$ ein Nenner von λ durch p teilbar ist. Der größte gemeinsame Teiler von diesem Nenner und n liefert dann einen echten Teiler von n. Das gleiche gilt natürlich auch für Vielfache k von N'. Enthält nun das N' für ein p nur 'kleine' Primfaktoren, so ist

$$k = \prod_{p \in P, p \leq B} p^{[\log C / \log p]}, \tag{9.6}$$

mit gewissen Schranken B und C, ein Vielfaches von N' (k ist das Produkt aller Primzahlpotenzen $\leq C$, die Potenzen von Primzahlen $\leq B$ sind). Der Versuch, kP zu bestimmen, führt dann sehr wahrscheinlich zu einem Teiler von n. Haben alle N' einen großen Primfaktor, so kann man kP (im Allgemeinen) ohne Probleme bestimmen. In diesem Fall kann man aber ein neues Paar (E, P) wählen.

Algorithmus 9.13 (ECFakt)

Eingabe: Ein zusammengesetztes $n \in \mathbb{N}$, eine Schranke A für die Anzahl der Versuche;
Ausgabe: Eventuell ein Teiler von n;

begin
  Wähle B und C;
  Setze $\mathcal{B} = \{p \in \mathcal{P} : p \leq B\} = \{p_1, \ldots, p_r\}$;
  Bestimme $l_1, \ldots, l_r$ mit $p_i^{l_i} \leq C < p_i^{l_i+1}$;
  for $t := 1$ **to** A **do**
  begin
    repeat
      wähle a, x, y zufällig;
      berechne $b = y^2 - x^3 - ax \bmod n$; $(P = (x, y) \in E)$
    until ggt $(4a^3 + 27b^2, n) = 1$;
    (* Nun Berechnung von kP *)
    $Q := P$;
    for $i := 1$ **to** r **do**
    for $j := 1$ **to** l_i **do**
    begin
      if alle Operationen zur Bestimmung von $p_i \cdot Q$ erlaubt **then** $Q := p_i \cdot Q$
      else erhalte Teiler von n; STOP;
    end;
  end;
  Meldung 'Verfahren erfolglos';
end

Bemerkung: (1) C gibt (grob gesprochen) eine Schranke für die Primteiler p an, die man mit obiger Methode finden kann. Zu gegebenem n kann man z. B. $C > (n^{1/4} + 1)^2$ setzen. Dann gilt mit dem Satz von Hasse

$$N' \leq p + 1 + 2\sqrt{p} = (\sqrt{p} + 1)^2 \leq (n^{1/4} + 1)^2 < C.$$

Wird N' dann durch keine Primzahl $> B$ geteilt, dann ist k ein Vielfaches von N'.
(2) Die Schranke B sollte einerseits nicht zu klein gewählt werden, damit die Wahrscheinlichkeit, daß N' aus solchen Primfaktoren zusammengesetzt ist, nicht zu klein wird. Andererseits sollte B nicht zu groß sein, da sonst die Berechnung von kP zu lange dauert.
(3) Bei geeigneter Wahl von B und C gehört das Verfahren von Lenstra mit zu den besten Faktorisierungsverfahren.
(4) Hat n einen Primfaktor, der wesentlich kleiner als $\sqrt{n}$ ist, so ist das EC–Faktorisierungsverfahren deutlich schneller als die Konkurrenten. Es kann in Kombination mit anderen Verfahren, bei denen die Faktorisierung als ein Hilfsproblem vorkommt, gebraucht werden.
(5) Das Verfahren von Lenstra benötigt weniger Speicherplatz als seine Konkurrenten.
(6) Ein interessanter Aspekt der Faktorisierung mit elliptischen Kurven ist, daß sich das Verfahren gut zur Parallelisierung eignet. Hat man mehrere Computer zur Verfügung, so kann man auf diesen das Verfahren mit verschiedenen elliptischen Kurven gleichzeitig laufen lassen.

Beispiel: Gegeben sei $n = 5429$. Zur Faktorisierung benutzen wir die Schar von elliptischen Kurven gegeben durch $y^2 = x^3 + ax - a$, die alle den Punkt $(1, 1)$ enthalten. Für jedes a, das wir benutzen wollen, ist zunächst zu prüfen, ob $\text{ggt}\,(4a^3 + 27a^2, n) = 1$.
Es sei $B = 3$ und $C = [(n^{1/4} + 1)^2] + 1 = 92$. Dann ergibt sich $l_1 = 6$ und $l_2 = 4$ und $k = 2^6 \cdot 3^4 = 5184$.
$a = 1$ ist erlaubt, da $(31, n) = 1$. Es wird nacheinander berechnet: $2P = (2, 5426)$, $4P = (2866, 2589)$, $8P = (875, 369)$, $16P = (2041, 1177)$, $32P = (168, 605)$, $64P = (4041, 813)$, $192P = (875, 5060)$, $576P = (649, 2438)$, $1728P = (1719, 3183)$ und $5184P = (2041, 4252)$ und alles funktioniert ohne Probleme.
Also gehen wir über zu $a = 2$. Wegen $(140, n) = 1$ ist dies erlaubt. Es wird nacheinander berechnet: $2P = (4076, 3384)$, $4P = (2435, 2775)$, $8P = (3743, 158)$, $16P = (3211, 2547)$, $32P = (688, 3174)$, $64P = (3804, 2892)$, $192P = (4279, 746)$.
Anschließend erhält man $2 \cdot 192P = (5316, 3768)$. Für die Addition von $192P$ und $2 \cdot 192P$ müßte man $5316 - 4279 = 1037$ modulo n invertieren. Dies geht aber nicht, wegen $\text{ggt}\,(1037, 5429) = 61$. Somit erhält man den Primteiler $p = 61$.
Mit $a = 3$ würde man den komplementären Teiler 89 finden.

10 Identifikation

Im alltäglichen Leben kann man seine Identität leicht nachweisen durch sein Aussehen, die Stimme, die Geburtsurkunde oder den Personalausweis (aber auch hierbei können Probleme auftreten, wie das Beispiel der Zarentochter Anastasia zeigt). Es gibt aber immer mehr Fälle, in denen man dies in elektronischer Form tun muß. Beispiele sind

- Einloggen bei einem Unix–Rechner.
- Zugang zu geschützten Internetseiten.
- Gebrauch der Kreditkarte.
- Abheben von Bargeld an einem Automaten.
- Homebanking.

Der Zugang zu Unix– und NT–Systemen funktioniert normalerweise über **Paßwörter**. Jeder berechtigte Benutzer gibt ein selbst gewähltes Paßwort w an. Im Rechner wird der Funktionswert $f(w)$ von w unter einer **Einwegfunktion** f gespeichert. Einwegfunktionen sind dadurch gekennzeichnet, daß sie leicht zu berechnen sind, aber praktisch nicht zu invertieren. Zum Beispiel kann der Wert eines Polynoms $f(x) \in \mathbb{F}_p[x]$ mit Grad n mit dem Horner Schema in einer Zeit von $O(\log^2 p \log n)$ berechnet werden (dies geht noch schneller, wenn das Polynom schwach besetzt ist). Für $p \geq 10^{20}$ und $n \geq 10^6$ ist die Berechnung der Wurzel des Polynoms zur Zeit nicht möglich. Es hat keinen Sinn, die Paßwortdatei auszuspionieren, da man aus den dort gespeicherten $f(w)$ die Paßwörter nicht rekonstruieren kann. Man sollte aber keine Wörter aus dem Duden auswählen, da ein Angreifer diese alle durchprobieren kann.
Eine Gefahr ist, daß das Paßwort über eine unsichere Leitung abgehört werden könnte. Dies kann man durch **Einmal–Paßwörter** umgehen. Neben einer persönlichen Geheimnummer ist bei jedem Vorgang (z. B. Überweisung) eine weitere Transaktionsnummer anzugeben. Dabei müssen natürlich vom **Beweiser** (z. B. Bankkunden) die Paßwörter $w_1, \ldots, w_n$ und vom **Verifizierer** (z. B. eine Bank) die Werte $f(w_1), \ldots, \ldots, f(w_n)$ geheim gehalten werden.

Challenge–Response–Identifikation

Bei den bisher beschriebenen Verfahren kann ein Angreifer lange vor seinem Betrugsversuch in den Besitz eines geheimen Paßworts kommen und es später einsetzen. Bei Challenge–Response–Verfahren bekommt der Beweiser B eine Aufgabe (Challenge), die er nur mit Hilfe eines Geheimnisses lösen kann. Diese Lösung (Response) schickt er an den Verifizierer V. V erkennt die Identität von B an, wenn die Lösung korrekt ist.
Im einfachsten Fall wird ein symmetrisches Verschlüsselungsverfahren verwendet. B und V kennen beide einen gemeinsamen Schlüssel K.

(1) B 'klopft an' bei V.

(2) V erzeugt eine Zufallszahl r und schickt sie an B.

(3) B verschlüsselt die Zufallszahl, bildet also $y = E_K(r)$.

(4) B schickt den Schlüsseltext y an V.

(5) V berechnet $r' = D_K(y)$.

(6) Im Fall $r' = r$ wird die Identität von B anerkannt.

Der Nachteil hierbei ist, daß auch V den geheimen Schlüssel kennen muß. Sicherer ist die Verwendung Public–Key–Signaturverfahren. Hier kann B als Antwort die Zufallszahl mit dem geheimen Schlüssel signieren und an V senden. V kann dann die Unterschrift mit dem öffentlichen Schlüssel von B verifizieren. Jedoch muß der öffentliche Schlüssel von B vor Manipulation von Dritten geschützt werden. Gelingt es einem Angreifer, den öffentlichen Schlüssel von B durch seinen zu ersetzen, so kann er sich erfolgreich als B ausgeben.

Zero–Knowledge–Beweise

Man kann auch spezielle Identifikationsverfahren benutzen. Als Beispiel hierfür betrachten wir Zero–Knowledge–Beweise (ZK–Beweise). B kennt ein Geheimnis, V nicht. In dem Protokoll überzeugt B V davon, daß er das Geheimnis kennt. Bei den Challenge–Response Verfahren könnte sich der Verifizierer V vielleicht eine geschickte Strategie ausdenken, um Teilinformationen über das Geheimnis des Beweisers B zu erhalten. Das besondere an ZK–Beweisen ist, daß man mathematisch beweisen kann, daß V nichts über das Geheimnis erfährt, obwohl er am Ende von der Identität von B überzeugt ist. Ein Protokoll hat die **Zero–Knowledge–Eigenschaft**, wenn es sich von V simulieren läßt, ohne daß er tatsächlich mit B kommuniziert, und die dabei auftretenden Datenströme nicht von denen des echten Protokolls zu unterscheiden sind. Durch ein echtes Protokoll kann V also nicht mehr Information erlangen, als er selber schon hat.

Das Fiat–Shamir Identifikationsprotokoll

Die Sicherheit des Verfahrens von Fiat–Shamir beruht auf der Schwierigkeit, Wurzeln modulo n zu ziehen, wenn die Faktorisierung von n nicht bekannt ist. Aus Abschnitt 5.3 ist bekannt, daß dies äquivalent zur Faktorisierung von n ist.

Vorab wählt B zwei Primzahlen p und q und berechnet $n = pq$. Dann wählt er ein zufälliges $s \in \{1, \ldots, n-1\}$ und berechnet $v = s^2 \bmod n$. Es sollte $(s, n) = 1$ sein, aber da man sonst einen Teiler von n gefunden hätte, ist dies so gut wie ausgeschlossen. Der öffentliche Schlüssel ist (v, n), der geheime Schlüssel ist s.

Will nun ein Beweiser B einem Verifizierer V seine Identität beweisen, so werden folgende Schritte durchgeführt:

(1) B wählt zufällig und gleichverteilt $r \in \{1, \ldots, n-1\}$ und berechnet $x = r^2 \bmod n$.

(2) B sendet x an V.

(3) V wählt zufällig und gleichverteilt ein $e \in \{0, 1\}$ und sendet e an B.

(4) Im Fall $e = 0$:

  (a) B schickt $y = r$ an V.

  (b) V prüft, ob $y^2 \equiv x \bmod n$.

(5) Im Fall $e = 1$:

  (a) B schickt $y = rs \bmod n$ an V.

  (b) V prüft, ob $y^2 \equiv xv \bmod n$.

Beispiel: Es sei $n = 391 = 17 \cdot 23$. Der geheime Schlüssel von B sei $s = 123$. Damit ist der öffentliche Schlüssel $(271, 391)$. Nun will B dem Verifizierer V seine Identität beweisen. Er wählt $r = 197$ und schickt $x = 197^2 \bmod 391 = 100$ an V. V wählt $e = 1$ und schickt es an B. B berechnet $y = 197 \cdot 123 \bmod 391 = 380$ und schickt es an V. V verifiziert $380^2 \equiv 121 \equiv 100 \cdot 271 \bmod 391$.

Mit Kenntnis von s ist B in der Lage, sowohl im Fall $e = 0$ als auch im Fall $e = 1$ die richtige Antwort zu geben (das Protokoll ist **vollständig**). Ein Betrüger kann sich auf den Fall $e = 0$ einstellen: er wählt r zufällig, berechnet x und kann dann $y = r$ angeben. Wenn er von $e = 1$ ausgeht, dann kann er y vorgeben und ein x bestimmen mit $x \cdot v \equiv y^2 \bmod n$. Allerdings muß er diese Auswahl treffen, bevor über e entschieden wird. Er ist nicht in der Lage, in beiden Fällen korrekt zu antworten. Denn wer r und $y = rs \bmod n$ kennt, der kann $s = y \cdot r^{-1} \bmod n$ berechnen. Diese Quadratwurzel kennt aber nur B. Ein Betrüger kann also nur mit Wahrscheinlichkeit $1/2$ ein korrektes Protokoll abliefern. Bei k Wiederholungen sinkt diese Wahrscheinlichkeit auf $1/2^k$, so daß dies für großes k praktisch ausgeschlossen ist (das Protokoll ist **korrekt**).

Bei dem Protokoll treten die folgenden Datenströme auf

(1) x ist gleichverteilt unter den Quadraten modulo n.

(2) e ist gleichverteilt in $\{0, 1\}$.

(3) y ist gleichverteilt in $\{1, \ldots, n-1\}$, da $(s, n) = 1$.

V kann das Protokoll wie folgt simulieren. Er wählt $y \in \{1, \ldots, n-1\}$ und $e \in \{0, 1\}$ zufällig und gleichverteilt. Er setzt $x = y^2 \cdot v^{-e} \bmod n$. Dann gilt wie im Originalprotokoll $y^2 \equiv x \cdot v \bmod n$. Ferner ist $x = (ys^{-e})^2 \bmod n$ unter den Quadraten modulo n gleichverteilt.

Das Identifikationsverfahren von Schnorr

Vorab werden folgende Parameter durch eine sogenannte **trusted authority** T bestimmt:

(1) Eine große Primzahl p ($p \geq 2^{512}$) mit der Eigenschaft, daß es in $\mathbb{F}_p^*$ 'unmöglich' ist, Diskrete Logarithmen zu berechnen.

(2) Ein großer Primteiler q von $p-1$ ($q \geq 2^{140}$).

(3) Ein Element $g \in \mathbb{F}_p^*$ mit Ordnung q (eine $((p-1)/q)$–te Potenz eines Erzeugers).

(4) Ein Sicherheitsparameter t mit $q > 2^t$ (z. B. $t = 40$).

(5) Ein sicheres, geheimes Signaturverfahren sig_T und das zugehörige (öffentliche) Verifikationsverfahren ver_T.

Ein Benutzer kann wie folgt ein **Zertifikat** von T erhalten:

(1) Die Identität wird gegenüber T auf konventionellem Wege (z. B. Geburtsurkunde, Personalausweis) belegt. Dann wird eine ID mit allen relevanten Daten angelegt.

(2) Der Benutzer wählt ein zufälliges $a \in \{0, \ldots, q-1\}$, berechnet $v = g^{-a} \bmod p$ und händigt v an T aus.

(3) T erzeugt eine Signatur $s = sig_T(\mathrm{ID}, v)$.

(4) Der Benutzer erhält das Zertifikat $C = (\mathrm{ID}, v, s)$.

Will nun ein Beweiser B einem Verifizierer V seine Identität beweisen, so werden folgende Schritte durchgeführt:

(1) B wählt ein zufälliges $k \in \{0, \ldots, q-1\}$ und berechnet $x = g^k \bmod p$.

(2) B sendet das Zertifikat $C(B) = (\mathrm{ID(B)}, v, s)$ zusammen mit x an V.

(3) V verifiziert die Signatur von T, indem er überprüft, daß $ver_T(\mathrm{ID(B)}, v, s)$ die Antwort 'korrekt' liefert.

(4) V wählt zufällig und gleichverteilt ein $r \in \{1, \ldots, 2^t\}$ und sendet r an B.

(5) B berechnet $y = k + ar \bmod q$ und schickt es an V.

(6) V verifiziert, daß $x \equiv g^y \cdot v^r \bmod p$.

Das Protokoll ist vollständig, da

$$g^y \cdot v^r \equiv g^{k+ar} \cdot v^r \equiv g^{k+ar} \cdot g^{-ar} \equiv g^k \equiv x \bmod p.$$

Beispiel: Es sei $p = 88667$, $q = 1031$ und $t = 10$. Das Element $g = 70322$ hat Ordnung q in $\mathbb{F}_p^*$. Der geheime Exponent von B sei $a = 755$. Somit ist $v = 70322^{1031-755} \bmod p = 13136$. Angenommen, B wählt $k = 543$. Dann berechnet er $x = 70322^{543} \bmod 88667 = 84109$ und sendet x an V. Angenommen V wählt $r = 1000$. Dann berechnet B $y = 543 + 755 \cdot 1000 \bmod 1031 = 851$ und schickt y an V. V verifiziert $84109 \equiv 70322^{851} \cdot 13136^{1000} \bmod p$ und akzeptiert die Identität von B.

Das Zertifikat wird deswegen verlangt, damit ein Betrüger sich nicht durch $C(\mathrm{ID(B)}, v', s')$ als B ausgeben kann und die dann folgende Aufgabe mit seinem a' lösen kann. Eine weitere Betrugsmöglichkeit wäre es, das echte Zertifikat von B zu verwenden (dieses ist bekannt, es wird ja bei jedem Protokoll übertragen). Dann wird aber das Original–v benutzt. Ohne das Original–a ist aber y aus r praktisch nicht zu bestimmen.

Das Schnorr–Verfahren ist korrekt:

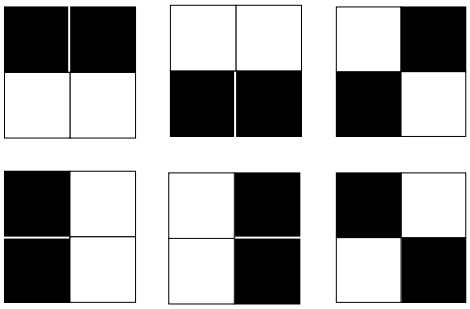
Satz 10.1 *Angenommen, ein Betrüger kennt einen Wert x, für den er mit Wahrscheinlichkeit $\epsilon \geq 1/2^{t-1}$ ein erfolgreiches Protokoll durchführen kann. Dann kann er a in polynomialer Zeit bestimmen.*

Also muß jeder, der ein Protokoll erfolgreich bestehen will, den geheimen Schlüssel a kennen.

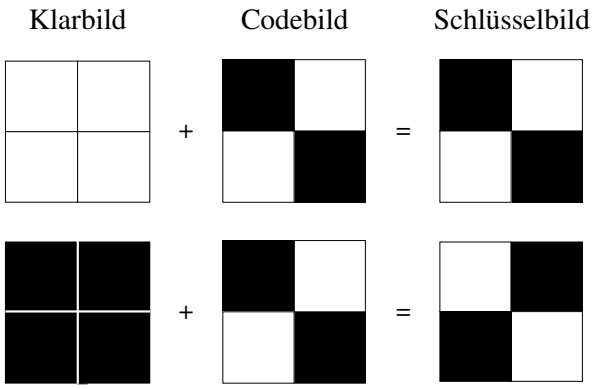
Bemerkung: (1) Es konnte bisher nicht gezeigt werden, daß das Schnorr–Verfahren sicher ist. Es könnte theoretisch möglich sein, durch geschickte Wahl der r Informationen über a zu erhalten.
(2) Es gibt eine Modifikation des Schnorr–Verfahrens von Okamoto, die unter gewissen Voraussetzungen sicher ist.
(3) Das Schnorr–Verfahren hat nicht die ZK–Eigenschaft.
(4) Der Speicherbedarf und der Rechenaufwand für den Beweiser sind relativ gering, so daß das Verfahren auf einer Smart Card implementiert werden kann (V hat einen leistungsstarken Computer zur Verfügung). Der größte Aufwand ist die Berechnung von $g^k \bmod p$. Nach jeder Identifikation können aber k und $g^k \bmod p$ schon 'auf Vorrat' berechnet werden.

11 Visuelle Kryptographie

Die Vigenère–Chiffre beruht darauf, daß zu einem Klartextblock ein gewisses Codewort addiert wird. Ein graphisches Analogon ist gegeben durch das Verfahren von Naor und Shamir. Dabei wird die Bildfläche in Großpixel unterteilt, von denen jedes aus 2×2 Pixeln besteht. Das 'visuelle Codewort' (oder 'Codebild') besteht aus Großpixeln mit je zwei weißen und schwarzen Pixeln. Dabei wird zufällig jeweils eine der sechs Möglichkeiten



gewählt. Jedes Großpixel des 'Klarbildes' wird in vier weiße bzw. vier schwarze Pixel zerlegt. Zur Verschlüsselung wird dann das Großpixel aus dem Codebild 'addiert'. Für jedes Pixel wird dabei eine Addition in $\mathbb{F}_2$ durchgeführt, wobei 'schwarz' der 1 und 'weiß' der 0 entspricht, z. B.



Durch Addition des Codebildes und des 'Schlüsselbildes' erhält man dann wieder das ursprüngliche Klarbild. Die Dekodierung ist sogar mit bloßem Auge möglich. Das Übereinanderlegen des Schlüsselbildes und des Codebildes ergibt das ursprüngliche Klarbild, jedoch mit einem Kontrastverlust. Dies kommt daher, daß beim Übereinanderlegen 'schwarz' + 'schwarz' wieder 'schwarz' und nicht 'weiß' ergibt.