Mathematische Grundlagen von öffentlichen Verschlüsselungsverfahren

Ralph–Hardo Schulz

Lehrerfortbildungsveranstaltung "Computersicherheit durch moderne Verfahren der Kryptologie"
14. März 2006

Es handelt sich hier um den 2.Teil einer Lehreinheit. Dabei wird von Vorkenntnissen in symmetrischen Verfahren und Anfangskenntnissen in öffentlichen Verfahren ausgegangen.

1 Regeln für modulares Rechnen

Vorerfahrung: z.B. Uhrzeit 15 Uhr $\overset{\text{mod } 12}{\longrightarrow}$ 3 Uhr (Nachmittag)

Wir verwenden folgende Definitionen (mit $n \in \mathbb{N}$ fest und $a, b \in \mathbb{Z}$):

1.1 Kongruenz modulo n

$$\begin{aligned} a \equiv b \pmod{n} \quad & (a \text{ kongruent } b \text{ modulo } n) \\ :\Longleftrightarrow\ & n \text{ teilt } a-b \quad (\text{in Zeichen } n \mid (a-b) \\ :\Longleftrightarrow\ & \exists\, t \in \mathbb{Z} : a - b = t \cdot n \\ \Longleftrightarrow\ & a \text{ und } b \text{ haben bei Division durch } n \text{ den gleichen Rest.} \\ \Longleftrightarrow\ & R_n(a) = R_n(b) \end{aligned}$$

Hierbei bezeichnet $R_n(m)$ den Rest zwischen 0 und $n-1$ von m bei Division durch n. Andere Bezeichnungen: $m \bmod n$, $m \operatorname{MOD} n$, $m\%n$.
Wir vermerken , dass u.a. $R_n(a) \equiv a \ (\bmod n)$ folgt. Es gelten folgende

1.2 Rechenregeln

$$\begin{aligned} a_i \equiv b_i \pmod{n} \text{ für } i = 1,2 \Rightarrow \quad a_1 + a_2 &\equiv b_1 + b_2 \pmod{n} \\ a_1 \cdot a_2 &\equiv b_1 \cdot b_2 \pmod{n} \\ a \equiv b \pmod{n} \Rightarrow \quad a^k &\equiv b^k \pmod{n}\,. \end{aligned}$$

Beispiel für das Rechnen mit Resten:

$$\begin{aligned} R_{12}(15) + R_{12}(11) = 3 + 11 = 14 &\equiv 2 (\bmod 12) \\ &\neq 2 \end{aligned}$$

Man beachte: $R_n(a_1 + a_2) = \boldsymbol{R_n}(R_n(a_1) + R_n(a_2))$!

Division einer Kongruenz durch t ist nur erlaubt, wenn t und n teilefremd sind:

$$\text{Für} \quad \text{ggT}(t,n) = 1 \quad \text{folgt} \quad [\, at \equiv bt \ (\bmod n) \ \Rightarrow \ a \equiv b \ (\bmod n)].$$

1.3 Restklassen:

Die Kongruenz mod n ist eine Äquivalenzrelation. Jede Äquivalenzklasse (*Restklasse*) ist von der Form $\bar{a} = \{a + kn \,|\, k \in \mathbb{Z}\} = a + n\mathbb{Z}$, besteht also aus allen ganzen Zahlen, die bei Division durch n den Rest a haben. Je zwei Restklassen mod n sind gleich oder disjunkt. Es gilt

$$\mathbb{Z} = \bigcup_{a=0}^{n-1} \bar{a}.$$

Beispiel(für $n = 2$):
$\mathbb{Z} = \overline{0} \ \cup \ \overline{1}$ für $\overline{0} = 2\mathbb{Z}$ und $\overline{1} = \{\ldots, -5, -3, -1, 1, 3, 5, \ldots\} = 2\mathbb{Z} + 1$.

2 Bestimmung der modularen Inversen

2.1 Teilen mit Rest

Zu $a, b \in \mathbb{N}$ mit $a > b$ existieren $q \in \mathbb{N}, r \in \mathbb{N} \cup \{0\}$ derart, dass

$$a = q \cdot b + r \quad \text{und} \quad r < b.$$

Hierbei ist $r = R_b(a)$ (in anderer Schreibweise: $a \mod b$ oder $r = a\%b$).
Anmerkung: Für den größten gemeinsamen Teiler gilt

$$\text{ggT}(a,b) = \text{ggT}(b,r) = \text{ggT}(b, R_b(a)).$$

Beweisskizze zur Anmerkung:
$d = \text{ggT(a,b)}$ teilt a und b, somit auch $r = a - q \cdot b$. Umgekehrt ist jeder Teiler von b und r auch Teiler von a. □

2.2 Euklidischer Algorithmus

Idee: Fortgesetzte Division mit Rest – bis der Rest 0 ist.
Ausführlich:

$$\begin{aligned}
a &= q_1 b + a_3 \\
b &= q_2 a_3 + a_4 \\
a_3 &= q_3 a_4 + a_5 \\
&\vdots \\
a_{\text{alt}} &= q \cdot a_{\text{mitte}} + a_{\text{neu}} \\
&\vdots \\
a_{n-1} &= q_{n-1} a_n + a_{n+1} \quad \text{mit } a_{n+1} \neq 0 \\
a_n &= q_n \cdot a_{n+1} + 0
\end{aligned}$$

Programmschritt: $\boxed{a_{neu} = a_{alt} \% a_{mitte}}$

Ergebnis: $a_{n+1} = \mathrm{ggT}(a, b)$

Beweisskizze: Der Rest 0 tritt schließlich wegen $b > a_3 > a_4 \ldots \geq 0$ auf. Ferner gilt

$$\begin{aligned} \mathrm{ggT}(a,b) &= \ldots = \mathrm{ggT}(a_{\mathrm{alt}}, a_{\mathrm{mitte}}) = \mathrm{ggT}(a_{\mathrm{mitte}}, a_{\mathrm{neu}}) = \ldots = \mathrm{ggT}(a_{n+1}, 0) \\ &= a_{n+1} \ . \end{aligned}$$

□

2.3 Erweiterter Euklidischer Algorithmus

Ziel: Auffinden von Zahlen $x, y \in \mathbb{Z}$ mit

$$\mathrm{ggT}(a, b) = x \cdot a + y \cdot b.$$

Diese existieren gemäß **Vielfachsummensatz** (Lemma von Bachet, 1624).

Idee: Bei jedem Schritt des Euklidischen Algorithmus ist der neue Rest Summe von Vielfachen von a und b, so dass man nur Buch zu führen braucht.

Beispiel:

$$\begin{array}{rcl} 60 &=& 2 \cdot 23 + 14 \\ & & \qquad\qquad 14 = 1 \cdot 60 - 2 \cdot 23 \\ 23 &=& 1 \cdot 14 + 9 \\ & & \qquad\qquad 9 = 23 - 14 \\ & & \qquad\quad = 23 - 60 + 2 \cdot 23 \\ & & \qquad\quad = -1 \cdot 60 + 3 \cdot 23 \end{array}$$

usw. ...

Ergebnis: $\mathrm{ggT}(60, 23) = 1 = 5 \cdot 60 + (-13) \cdot 23.$

Rekursion (s. auch Figur 2.1): Startwerte: $a = 1 \cdot a + 0 \cdot b$ und $b = 0 \cdot a + 1 \cdot b$. Sei $a_{\mathrm{alt}} == x_{\mathrm{alt}} a + y_{\mathrm{alt}} b$ und $a_{\mathrm{mitte}} = x_{\mathrm{mitte}} a + y_{\mathrm{mitte}}\, b$, ! Dann folgt

$$\begin{aligned} a_{\mathrm{neu}} &= a_{\mathrm{alt}} - q \cdot a_{\mathrm{mitte}} \\ &= x_{\mathrm{alt}} \cdot a + y_{\mathrm{alt}} \cdot b - q(x_{\mathrm{mitte}} \cdot a + y_{\mathrm{mitte}} \cdot b) \\ &= \underbrace{(x_{\mathrm{alt}} - q \cdot x_{\mathrm{mitte}})}_{x_{\mathrm{neu}}} a + \underbrace{(y_{\mathrm{alt}} - q \cdot y_{\mathrm{mitte}})}_{y_{\mathrm{neu}}} \cdot b. \end{aligned}$$

Daher setzt man:

$$\boxed{x_{\mathrm{neu}} = x_{\mathrm{alt}} - q \cdot x_{\mathrm{mitte}} \ \text{ und } \ y_{\mathrm{neu}} = y_{\mathrm{alt}} - q \cdot y_{\mathrm{mitte}} \quad \text{für} \quad q = \left\lfloor \frac{a_{\mathrm{alt}}}{a_{\mathrm{mitte}}} \right\rfloor}$$

2.4 Berechnung der Inversen

Eine Inverse zu a modulo b , also ein $x \in \{1, \ldots n-1\}$ mit

$$(*) \qquad\qquad x \cdot a \equiv 1 \ (\bmod n),$$

```
def Bachet(a, b):
    aalt, amitte = a, b
    xalt, xmitte = 1, 0
    yalt, ymitte = 0, 1
    while amitte <> 0 :
        q= aalt / amitte
        aneu= aalt % amitte
        xneu= xalt - xmitte *q
        yneu= yalt - ymitte *q
        xalt, xmitte = xmitte, xneu
        yalt, ymitte = ymitte, yneu
        aalt, amitte = amitte, aneu
    return aalt, xalt, yalt
```

Ausgabe: $\mathrm{ggT}(a,b), x, y$ (mit $\mathrm{ggT}(a,b) = x \cdot a + y \cdot b$).

Figur 2.1: Erweiterter Euklidischer Algorithmus, **Algorithmus von Berlekamp** zur Berechnung von $\mathrm{ggT}(a,b)$ und von x, y mit $\mathrm{ggT}(a,b) = x \cdot a + y \cdot b$. (Python-Programm nach H. Witten 2001)

existiert, falls $\mathrm{ggT}(a,n) = 1$ ist. In letzterem Falle gibt es nach Teil 2.3 Zahlen x_1, y_1 mit $x_1 \cdot a + y_1 \cdot n = 1$. Durch Reduktion der Gleichung modulo n ergibt sich daraus $R_n(x_1) \cdot a + 0 \equiv 1 (\bmod n)$; also ist $x = R_n(x_1)$ zu a invers modulo n.
Algorithmus: s. Figur 2.2 !

```
n=input('Modul eingeben: ')
while 1: # Endlosschleife
      a = input('Welche Zahl soll invertiert werden
      (Ende mit 0)? ')
      if a == 0: break
      ggT, x, y =Bachet (m, a)
      if ggT== 1:
          if y < 0 : y = y + m
          print 'Das modulare Inverse zu ',a,'ist',y
      else: print a, 'ist nicht invertierbar (ggT=',ggT,') !'
print; print 'Fertig!'; print
```

Figur 2.2: Python-Programm zur Berechnung von modularen Inversen (nach H. Witten 2001)

3 Weitere algebraische Grundlagen des RSA–Kryptoverfahrens

Thema:
Anwendung des erweiterten Euklidischen Algorithmus und des Satzes von Fermat beim RSA–Verfahren

3.1 Kleiner Satz von Fermat

Für eine Primzahl p und alle zu p teilerfremden ganzen Zahlen a gilt

$$\boxed{a^{p-1} \equiv 1 \pmod{p}}\ .$$

Beweis (durch Betrachtung der "Restklassenvertreter"): Die möglichen Reste ungleich 0 bei Division einer Zahl durch p sind $1, 2, 3, \ldots, p-1$. Die Zahlen $1 \cdot a, 2 \cdot a, 3 \cdot a, \ldots, (p-1) \cdot a$ sind ebenfalls $p-1$ verschiedene Zahlen; je zwei von ihnen haben verschiedene Reste (ungleich 0) $\mod p$, da wegen $\mathrm{ggT}(a, p) = 1$ gilt:

$$k \cdot a \equiv l \cdot a (\bmod p) \Longrightarrow k \equiv l \,(\bmod\, p),$$

was wegen der Größe dieser Zahlen nur für $k = l$ möglich ist. Also ist jede der Zahlen $i \cdot a$ kongruent zu genau einem Element aus $\{1, 2, 3, \ldots, p-1\}$, und es gilt

$$\begin{array}{rcll} \overline{1 \cdot a} \cdot \overline{2a} \cdot \overline{3a} \cdot \ldots \overline{(p-1)a} & = & \bar{1} \cdot \bar{2} \cdot \bar{3} \cdot \ldots \overline{(p-1)} & \text{d.h.} \\ (1 \cdot 2 \cdot 3 \cdot \ldots \cdot (p-1)) \cdot a^{p-1} & \equiv & 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (p-1) & (\text{m mod } p\ ); \end{array}$$

hieraus folgt durch Division $a^{p-1} \equiv 1 \pmod{p}$. □

3.2 Anmerkung: Satz von Euler

Für beliebige Moduln n gilt die folgende Verallgemeinerung, der Satz von Euler:

Für jede ganze Zahl a, die zu $n \in \mathbb{N}$ teilerfremd ist, gilt

$$\boxed{a^{\varphi(n)} \equiv 1 \pmod{n}}$$

Hierbei bezeichnet $\varphi(n)$ die Anzahl der zu n teilerfremden Zahlen aus der Menge $\{1, \ldots, n-1\}$; z.B. ist $\varphi(p) = p - 1$ für jede Primzahl p und, hier wichtig,

$$\varphi(p \cdot q) = (p-1)(q-1)$$

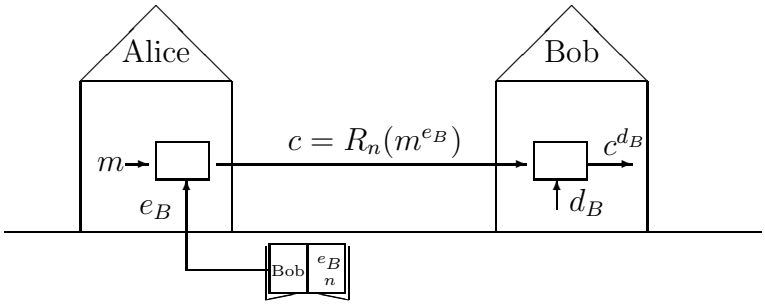
für voneinander verschiedene Primzahlen p, q.

3.3 Verifizierung des RSA-Verfahrens durch Anwendung des Satzes von Fermat

Korrektheit der Entschlüsselung: Beim RSA–System mit öffentlichen Schlüsseln n, e und geheimem Schlüssel d sind diese wie folgt gewählt:

$$\boxed{\begin{array}{ll} n = p \cdot q & \text{für zwei Primzahlen } p, q \\ d \cdot e \equiv 1 & (\mod (p-1)(q-1)) \end{array}}$$

Insbesondere existiert ein t mit $e \cdot d = t(p-1)(q-1)+1$. Für Zahlen m kleiner n (als Nachrichten) wird die Chiffrierung gemäß Figur 3.1 vorgenommen



Figur 3.1: Schema der Chiffrierung und Dechiffrierung beim RSA–System

Wir zeigen nun, dass c^d den Klartext ergibt.
(Erinnerung: $R_n(b)$ ist der Rest zwischen 0 und $n-1$ von b bei Division durch n).
Aus $c = R_n(m^e)$ folgt $c \equiv m^e \pmod n$ und damit $c^d \equiv m^{ed} \pmod n$.
Ist m zu p teilerfremd, so gilt

$$m^{ed} = m^{t(p-1)(q-1)+1} = (m^{(t(q-1)})^{p-1} \cdot m$$

und daher nach dem Satz von Fermat $m^{ed} \equiv 1 \cdot m \equiv m \pmod p$.
Ist m nicht zu p teilerfremd, so folgt $m^{ed} \equiv 0 \equiv m \pmod p$. Also erhält man in beiden Fällen $m^{ed} \equiv m \pmod p$.
Analog ergibt sich $m^{ed} \equiv m \pmod q$. Insgesamt folgt, dass sowohl p als auch q und damit auch ihr Produkt $p \cdot q$ Teiler von $m^{ed} - m$ sind. Also gilt $m^{ed} - m \equiv 0 \pmod{pq}$ und folglich $R_n(c^d) = m$. □

3.4 Anwendung des erweiterten Euklidischen Algorithmus beim RSA–Verfahren

Bei der Schlüsselvergabe sind Zahlen e und d zu erzeugen, die bezüglich des Moduls $(p-1)(q-1)$ invers zueinander sind, für die also gilt:

$$e \cdot d \equiv 1 \pmod{(p-1)(q-1)}.$$

Wählen wir eine natürliche Zahl e mit $\text{ggT}((p-1)(q-1), e) = 1$, so lassen sich nach dem Vielfachsummensatz (s. 2.4) Zahlen x und y aus $\mathbb{Z}$ finden mit

$$x \cdot (p-1)(q-1) + y \cdot e = 1.$$

Modulo $(p-1)(q-1)$ geht diese Gleichung über in

$$0 + y \cdot e \equiv 1 \,(\text{mod}(p-1)(q-1)),$$

so dass also y eine Modulare Inverse von e ist und $d := y$ gesetzt werden kann.
Beispiel (mit für kryptographische Zwecke zu kleinen Zahlen) :
Seien $p = 7$ und $q = 11$; dann ist $(p-1)(q-1) = 60 = a$; dazu ist $b = e = 23$ teilerfremd und kann als öffentlicher Schlüssel gewählt werden. Wegen $1 = 5 \cdot 60 + (-13) \cdot 23$ folgt

$$1 \equiv (-13) \cdot 23 \ (\text{mod } (p-1)(q-1));$$

also ist $d = 60 - 13 = 47$ der geheime Schlüssel zu $e = 23$.

3.5 Primzahltests

Primzahltests und Faktorisierung sind Themen von großer Bedeutung für das RSA-Verfahren (zur Schlüsselgenerierung und für die Sicherheitsanalyse). Auf den ersten Blick scheint die Frage, ob eine Zahl eine Primzahl ist, gleichwertig mit der Suche nach Teilern zu sein. Oft kann man aber ohne Bestimmung der Teiler zeigen, dass eine Zahl zusammengesetzt ist. Wir behandeln hier zwei dieser Verfahren.

3.5.1 Fermat-Test und Charmichael-Zahlen

Ist p eine Primzahl, so gilt für jede zu p teilerfremde Zahl a nach dem kleinen Satz von Fermat, s. (3.1), die Kongruenz

$$a^{p-1} \equiv 1 \,(\,\text{mod}\, p).$$

Ist n eine zu testende Zahl, und findet man eine Zahl $a < n$ mit $a^{n-1} \not\equiv 1(\,\text{mod}\ n)$, so folgt daraus, dass n keine Primzahl ist. Wenn hingegen $a^{n-1} \equiv 1(\,\text{mod}\, n)$ ist, so sagt man, n habe den Fermat-Test mit Testbasis a bestanden.
Es gibt Zahlen, die den Fermat-Test bestehen, ohne dass sie Primzahl sind; solche Zahlen nennt man **Pseudoprimzahlen**. 341 ist ein *Beispiel* mit Basis 2 (s.z.B. DIFF Studienbrief oder Buchmann, s.Literaturliste am Ende des Paragraphen!).
Es gibt sogar Zahlen n, die für jede zu n teilerfremde Zahl als Basis den Fermat-Test bestehen und trotzdem keine Primzahlen sind. Sie heißen **Carmichael-Zahlen**. *Beispiel:* 561 ist die kleinste Carmichael-Zahl.

3.5.2 Miller-Rabin-Test

Sei $n = 2^s \cdot m + 1$ mit m ungerade. Man sagt nun, dass n den Miller-Rabin-Test mit Testbasis a (für $1 < a < m$) besteht, wenn gilt:

$$\boxed{a^m \equiv 1 \,(\bmod n) \;\text{ oder }\; a^{2^j m} \equiv -1 \,(\bmod n) \text{ für ein } \quad j \in \{0, 1, \ldots s-1\}.}$$

Miller-Rabin-Test

Es gilt der folgende
Satz: *Jede Primzahl besteht den Miller–Rabin–Test.*
Beweis: Sei $p = 2^s \cdot m + 1$ Primzahl und $a^m \not\equiv 1 (\bmod p)$. Nach dem kleinen Satz von Fermat, s. (3.1), gilt aber zumindest $a^{p-1} = a^{2^s m} \equiv 1 (\bmod p)$. In der endlichen Folge $(a^m, a^{2m}, a^{2^2 m} \ldots, a^{2^s m})$ ist das letzte Glied also kongruent 1, eventuell aber schon ein vorangehendes Glied (und jedes nachfolgende), nicht aber das erste. Sei also

$$x^2 = a^{2^i m} \equiv 1 \,(\bmod p)$$

und i minimal mit dieser Eigenschaft; es ist p Teiler von $x^2 - 1 = (x+1)(x-1)$ und als Primzahl Teiler von $x - 1$ oder $x + 1$. Da i minimal ist, folgt $x \not\equiv 1 \,(\bmod p)$ und folglich $a^{2^{i-1} m} = x \equiv -1 \,(\bmod p)$. □
Beispiel: Die Carmichael Zahl $n = 561$ besteht wegen $2^{35} \equiv 263 \,(\text{mod } 561)$ und $2^{140} \equiv 67 (\text{mod } 561)$ sowie $2^{280} \equiv 1 (\text{mod } 561)$ nicht den Miller–Rabin–Test.
Anmerkungen:

1. Der Miller–Rabin–Test ist strenger als der Fermat–Test; denn jede Zahl, die den Miller–Rabin–Test besteht, passiert auch den Fermat–Test erfolgreich: Aus $a^m \equiv 1 \,(\bmod m)$ folgt $a^{n-1} = (a^m)^{2^s} \equiv 1 (\bmod n)$; aus $a^{2^j m} \equiv -1 \,(\bmod n)$ für $j < s$ ergibt sich

$$a^{n-1} = (a^{2^j m})^{2^{s-j}} \equiv (-1)^{2^{s-j}} \equiv 1 \,(\bmod n).$$

2. M.O.Rabin konnte zeigen, dass eine zusammengesetzte Zahl bei mindestens $\frac{3}{4}$ der Zahlen zwischen 1 und $n-1$ als Basen den Miller–Rabin–Test nicht besteht. Durch k Tests und zufällige Wahl der Basen kann man mit Wahrscheinlichkeit größer gleich $1 - \left(\frac{1}{4}\right)^k$ eine zusammmengesetzte Zahl als solche erkennen.

3.6 Faktorisierung

3.6.1 Rechenzeit beim einfachsten Verfahren

Ein naives Verfahren zur Faktorisierung einer Zahl n ist die Division von n durch alle Zahlen kleiner gleich $\sqrt{n}$. Bei einer Zahl mit 100 Dezimalstellen müssten aber alle Primzahlen kleiner 10^{50}, d.h. ca $8 \cdot 10^{47}$ Primzahlen, getestet werden (s.DIFF).

3.6.2 Quadratisches Sieb

Effektivere Verfahren zur Faktorisierung von n benutzen das "Quadratische Sieb ", bei dem Zahlen $x, y \in \mathbb{Z}$ mit

$$x^2 \equiv y^2 \,(\bmod n) \quad \text{und} \quad x \not\equiv \pm y \,(\bmod n)$$

bestimmt werden. Es gilt dann: n teilt $x^2 - y^2$, aber nicht $x - y$ und $x + y$; es existiert dann eine ganze Zahl t mit $(x - y)(x + y) = tn$; wären $(x - y)$ und n teilerfremd, so n ein Teiler von $x + y$, ein Widerspruch; somit hat man mit $\mathrm{ggT}\,(x - y, n)$ einen echten Teiler von n gefunden hat. (Weiteres s. z.B. Buchmann).

3.6.3 Anmerkungen zur Faktorisierung

Auch diese und andere (zum Teil probabilistische) Faktorisierungs–Algorithmen stoßen schnell an ihre Grenzen, z.B. das "General Number Field Sieve (GNFS)" oder die "Elliptische–Kurven–Methode (ECM)" bei Zahlen mit zur Zeit ca 160 Dezimalstellen.
Um etwas nachempfinden zu können, wie sehr der Zeitaufwand zur Faktorisierung des Moduls n (und damit der Hauptschritt zur unbefugten Entschlüsselung) von RSA mit steigender (Dezimal-) Stellenzahl der Faktoren p und q anwächst, sollten Sie Zahlen mit immer größer werdender Stellenzahl (und ohne kleine Primteiler) dem Faktorisierungsprogramm eines Computeralgebrasystems, z.B. von Derive, Mathematica oder Matlab, eingeben. Notieren Sie jeweils die Stellenzahl der Zahl und des kleinsten Faktors sowie die Rechenzeit! Beobachten Sie den Anstieg der letzteren bei steigender Stellenzahl!

Literatur zu §3:

Buchmann, J.: Einführung in die Kryptographie. Springer Verlag, Wiesbaden, Heidelberg, 1999, 2001[2].

CrypTool-Skript: Mathematik und Kryptographie.
http://www.CrypTool.de

Niederdrenk-Felgner: Algorithmen der elementaren Zahlentheorie. DIFF – Studienbriefe CM1, Tübingen 1988.

Bartholomé et al.: Zahlentheorie für Einsteiger, Vieweg Verlag, Braunschweig 1996.

R.-H. Schulz: Primzahlen in öffentlichen Chiffrierverfahren. Mathematik lehren 61 (1993)56-64.

R.-H. Schulz: Codierungstheorie. Eine Einführung. Kap.19. Vieweg Verlag, Wiesbaden 2003[2].

4 Cybergeld

4.1 Anforderungen an Zahlungsmittel

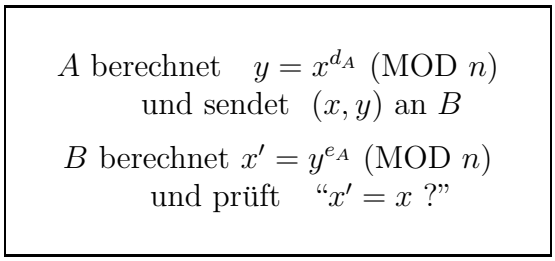
"Wer heute seine Kreditkartendaten in ein normales WWW-Formular einträgt und abschickt, ist selber schuld." (Hacker-Zitat, cf. PC-Magazin). Kreditkartendaten haben eine leicht erkennbare und automatisch aus Datenströmen kopierbare Struktur.
Wie bei den klassischen Zahlungsmitteln *Banknoten/ Münzen, Schecks* und *Gutscheinen* (zu letzteren rechnen z.B. Fahrkarten und Briefmarken) sind auch beim elektronischen Bezahlen **Fälschungssicherheit** (Nicht – Duplizierbarkeit, Nicht–Veränderbarkeit) und **Authentizität** (Ausgabe durch eine Notenbank, Unterschrift bei Schecks) unabdingbar. Erwünscht, aber nicht generell vorhanden, sind **Anonymität, Offenheit**(Übertragbarkeit) und **Verbindlichkeit** (– d.h. dass erfolgte Zahlungen nicht abgestritten werden können).
Auch bei den elektronischen Zahlungsmitteln unterscheidet man u.a. drei Typen.

- Elektronische Schecks
- Elektronisches Geld
- Elektronische Gutscheine

Auch virtuelle Kreditkarten sind im Gebrauch (Firma "First virtuell").

4.2 Digitale Signatur/ Elektronische Schecks

Hierbei werden Zahlungsanweisungen durch eine kryptographische **Signatur** geschützt. Dies muss so geschehen, dass die Signatur nicht ohne einen geheimen Schlüssel erzeugt werden kann, jede Änderung der Daten auch eine Änderung der Signatur erzeugt und eine Verifikation des Textes anhand der Signatur stattfinden kann.
Bei symmetrischen Verfahren (z.Bsp. DES) ist dazu der geheime Schlüssel (und damit eine vertrauenswürdige dritte Instanz, VDI) erforderlich, bei öffentlichen Verfahren ist die Signatur mittels des öffentlichen Schlüssels nachprüfbar.
Die Gefahr von ungedeckten Schecks lässt sich u.a. durch Betragsbeschränkungen und Nachfrage-Pflicht des Händlers, der den Scheck akzeptieren soll, bei einer Autorisierungszentrale reduzieren. Das Verfahren ist also nicht anonym.
In der Praxis wird der Text mittels einer sogenannten *Hash–Funktion* zunächst auf eine wesentlich kürzere Zeichenfolge reduziert und diese verschlüsselt an den Text der Scheckdaten angehängt.
Bei Benutzung des **RSA–Signiersystems** gehen Scheckaussteller A und Händler oder Autorisierungszentrale B vor, wie in Figur 4.1 angegeben. An-



Figur 4.1: Signatur mittels des RSA–Verfahrens
(Dabei bezeichnet (e_A, n) den öffentlichen, d_A den geheimen Schlüssel von A.)

wendungsbeispiele für dieses Modell sind die **"Point of Sale" (POS)** – Systeme, an deren Terminals der Kunde die Verschlüsselung durch die ihm als Teilnehmer zugeteilte Chip-Karte automatisch vornimmt.
Bei dem System von **Cyber Cash** wurden die Kreditkartendaten des (Internet-) Kunden durch einen CyberCash Server mit dem RSA–Verfahren verschlüsselt und elektronisch an den Händler gesandt, der seine Daten hinzufügt und an die Firma Cyber Cash GmbH (die inzwischen von der Firma VeriSign übernommen wurde) weiterleitet. Diese entschlüsselt die Daten und kontrolliert bei dem Kreditkarteninstitut die Gültigkeit der Karte und die Deckung. Erst danach erfolgt die Abbuchung. (Das Verfahren wurde von der Commerzbank und der Dresdner Bank angewandt).

4.3 Anonymes Digitales Geld

Das wichtigste Verfahren hierbei, das **E-Cash** der Firma DigiCash, jetzt bei ECash Direct, beruht auf einem Modell von D.Chaum (siehe Figur 4.2). (Es war u.a. ein Projekt der Deutschen Bank).

> **Anmerkung am 31.7.2012: Das Verfahren hat sich am Markt nicht durchgesetzt,** und die Firma DigiCash ging 1998 Bankrott. Ein anderes Protokoll, nämlich *"Bitcoin"*, das eine gewisse Verbreitung gefunden hat, ist noch aktuell.
> vgl. P.Wilhelm: Ideales elektronisches Geld. LogIn Nr.171,(2011/2012) p.25-30. und
> Jürgen Müller: Bitcoins - Geld selbst drucken. LogIn Nr.171 (2011/2012), p. 31-36.

Mittels einem auf seiner Festplatte gespeicherten Programm stellt der Kunde das Rohmaterial W für eine elektronische Münze her, z.Bsp. durch Codierung einer Zufallszahl. Wichtig ist, dass W ein wiedererkennbares "Redundanzschema" aufweist, z.B. durch Wiederholung in der Form *abcabc* oder *abccba*. Um die spätere Anonymität der Münze zu garantieren, multipliziert

der Kunde W noch mit einem "Blinding"-Faktor C^e, der Potenz einer Zufallszahl C. (Hierbei sei (e, n) ein öffentlicher und d der bei der Bank deponierte zugehörige geheime Schlüssel im RSA-System). (Mechanisches Modell: Die Nachricht wird in ein versiegeltes Couvert gelegt).
Die Bank zertifiziert $S = C^e \cdot W$ (MOD n) durch ihre Signatur $T := \text{sig}(S) := S^d$ (MOD n) (blinde Unterschrift; **blinde Signatur**); (als mechanisches Modell dient das folgende: durch eingelegtes Kohlepapier drückt sich die auf dem Couvert geleistete Unterschrift auf das nicht sichtbare Dokument durch.) Die Bank belastet das Konto des Kunden und sendet T an ihn; der Kunde besitzt nun mit

$$M := C^{-1} \cdot T \equiv C^{-1} S^d \equiv C^{-1} C^{ed} W^d \equiv W^d (\mod n)$$

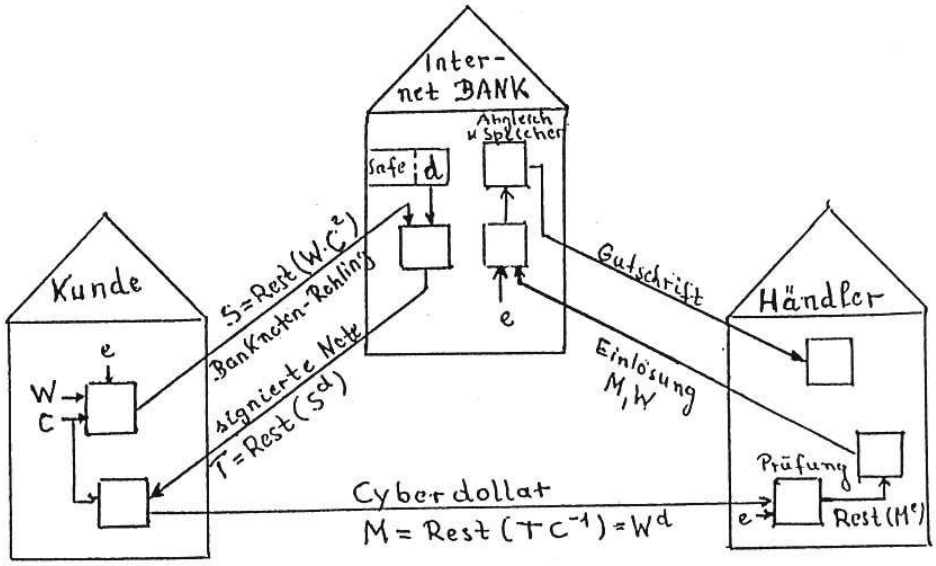
eine anonyme elektronische Münze, einen **Cyberdollar**.
Die Münze M lässt sich durch Bildung von $M^e \equiv W^{de} \equiv W (\text{mod } n)$ und Prüfung, ob W in das Redundanzschema passt, verifizieren.
Um das **Duplizieren einer Münze** zu verhindern, sind mehrere Methoden dienlich. Bei einem dieser Verfahren speichert die Notenbank alle zur Zahlung eingereichten Münzen (oder deren schon mit der Kundensoftware erzeugte Seriennummer). Dieses sehr aufwändige Verfahren akzeptiert die Münze nur vom ersten Einreicher; damit ist ein online-Verfahren zwischen Händler und Bank unumgänglich.
Bei einem alternativen Modell enthält das Redundanzschema der Münze auch einen Zusatz m von Daten über die Identität des Kunden. Dieser wählt Schlüssel $k_1, \ldots, k_r$ und erhält (evtl. nach sogenanntem Hashen des Textes) Geheimtexte $c_1 = f(k_1, m), \ldots, c_r = f(k_r, m)$; die Werte k_i und c_i lässt er von der Bank blind signieren. Der Verkäufer wählt dann r Bits $b_1, \ldots, b_r$ aus und lässt den Kunden bei $b_i = 1$ den Wert $\text{sig}(c_i)$, bei $b_i = 0$ den Wert $\text{sig}(k_i)$ mitteilen.
Wenn die Bank feststellt, dass die Münze schon einmal eingereicht wurde, dann haben die beiden Verkäufer mit Wahrscheinlichkeit $1 - (1/2)^r$ nicht dieselben Bitstrings gewählt. In diesem Fall gibt es ein i, für das die Bank sowohl c_i als auch k_i kennt. Dann kann sie m und damit die Identität des Kunden ermitteln.

4.4 Elektronische Gutscheine

Ein elektronischer Gutschein ist von der Form $G = (G_0, G_Z)$, wobei G_0 die grundlegenden Daten wie Betrag und Datum angibt und G_Z die kryptographische Zusatzinformation.
Als Praxisbeispiel sind vorausbezahlte **Telefonkarten** zu erwähnen.

4.5 Anmerkungen

- Bei vielen Anwendungen geht es nicht um elektronische Bezahlung, sondern nur um eine sichere Teilnehmer-Identifizierung. Beispiele: Mo-



Figur 4.2: Zum E-Cash - Verfahren

biltelefonnetze, Telefonieren mit Abbuchung vom Konto.

- Um Wiedereinspielung abgefangener Nachrichten zu verhindern, sind bei vielen Homebanking-Vorgängen vorher ausgegebene **Transaktionsnummern** "TAN"s in Verwendung.

Literatur zu §4:

Baumann, Rüdiger : Digitales Geld. Bestellen und Bezahlen im Internet. LOG IN **17**/2 (1997) 30-38.

Beutelspacher, Albrecht, T.Hueske und A.Pfau: Kann man mit Bits bezahlen? Informatik–Spektrum **16** (1993) 99-106.

Müller, Jürgen: Bitcoins. Geld selbst drucken. LogIn Nr.171 (2011/2012), p. 31-36. (Zusatz am 31.7.2012)

Weis, Rüdiger: Zahlen Sie bar oder digital? Protokolle für sicheres Geld. PC Magazin Spezial 5.98 (Kryptographie und Netzsicherheit) 1998, p. 94-97.

Schneppe, Ulrich: http://ulrich.schneppe.bei.t-online.de/s1916/teil61.htm

Wilhelm, Paul: Ideales elektronisches Geld. LogIn Nr.171, p.25-30. (2011/2012) (Zusatz am 31.7.2012)

5 Alternativen zu RSA

5.1 Diskreter Logarithmus statt Primfaktorzerlegung

Statt die Bildung des Produkts von zwei großen Primzahlen als Einwegfunktion zu verwenden, kann man auch das Potenzieren in einer (endlichen) abelschen Gruppe (z.B. der multiplikativen Gruppe von $\mathbb{Z}_p^* = \{1, \ldots, p-1\}$)

heranziehen, solange die Berechnung der Umkehrung praktisch unmöglich ist. Sei also G eine endliche kommutative Gruppe und $g \in G$. Wichtig ist dann die von g erzeugte zyklische Unterguppe

$$H =< g >= \{g^i \,|\, i \geq 0\} = \{g, g^2, g^3, \ldots, g^m\};$$

dabei bezeichnet $m = o(g)$ die Ordnung von g, also die kleinste natürliche Zahl m mit $g^m = 1$. Es folgt dann auch $|H| = m$ für die Ordnung (Elementeanzahl) der von g erzeugten Gruppe H.
Oft wählt man g speziell als erzeugendes Element von $G = \mathbb{Z}_p^*$.
Beispiel: $p = 11$

$$H =< 2 >: \quad 2, 4, 8, 16 \equiv 5, 10, 20 \equiv 9, 18 \equiv 7, 14 \equiv 3, 6, 12 \equiv 1$$

und $|G| = |H| = |\mathbb{Z}_{11}{}^*| = 11 - 1$.
Nicht jedes Element erzeugt $\mathbb{Z}_{11}{}^*$: $\quad | < 4 > | = 5 \neq 10$.
Als Einwegfunktion kommt die "diskrete Exponentialfunktion" in Frage:

$$\{1, 2, \ldots, |H| - 1\} \longrightarrow H \quad \text{mit} \quad k \mapsto g^k.$$

Beispiel: Wertetabelle der Exponentialfunktion im Fall $G = \mathbb{Z}_{11}^\star = H$ und $g = 2$:

k	1	2	3	4	5	6	7	8	9	10
$2^k (\text{mod} 11)$	2	4	8	5	10	9	7	3	6	1

Beim Problem der Umkehrung sind G, g und $g^k \in G$ gegeben, und k ist gesucht. Die zugehörige Funktion heißt "**diskrete Logarithmusfunktion**" und wird mit $\log_g$ bezeichnet.
Beispiel: Wertetabelle der Logarithmusfunktion im Fall $G = \mathbb{Z}_{11}^\star = H$ und $g = 2$:

$2^k (\text{ mod } 11)$	1	2	3	4	5	6	7	8	9	10
$k = \log_2(2^k)$	10	1	8	2	4	9	7	3	6	5

Die Berechnung des diskreten Logarithmus bei geeigneter Wahl von H und g ist viel schwieriger als das Potenzieren von g in G. Es gibt zwar Verfahren, die effizienter sind als einfaches Durchprobieren(s.u.); aber auch sie versagen oft wegen zu langer Rechenzeit.

5.2 Das ElGamal–Verfahren

Das "ElGamal–Verfahren" ist ebenfalls ein Public key–Verfahren, dessen Sicherheit aber darauf beruht, dass der diskrete Logarithmus im gegebenen Szenario nicht berechenbar ist. Das ElGamal-Verfahren im engeren Sinne

benutzt die multiplikative Gruppe GF $(q)^*$ "des" endlichen Körpers mit q Elementen. Stattdessen kann man aber eine beliebige endliche zyklische Gruppe verwenden. Wir behandeln hier diese allgemeinere Form.

Voraussetzungen: Seien

- G eine endliche Gruppe, $g \in G$ und $H :=< g >$ derart, dass die Berechnung des diskreten Logarithmus $\log_g$ in $H =< g >$ fast unmöglich ist,
- a eine natürliche Zahl mit $1 < a < |H|$ und
- $h := g^a$.

Die Ver- bzw. Entschlüsselung erfolgt wie in Figur 5.1 angegeben.

Öffentlicher Schlüssel	(g, h) (für $h := g^a$)
Privater Schlüssel	a
Verschlüsselung des Klartexts m	$E(m) := (g^k, m \cdot h^k)$ für ein zufällig gewähltes $k \in \{2, \ldots, \|H\| - 1\}$
Entschlüsselung	$D(x, y) := (x^a)^{-1} \cdot y$

Figur 5.1: (Verallgemeinertes) ElGamal–System
Verifizierung: Es gilt

$$D(g^k, mh^k) = ((g^k)^a)^{-1} mh^k = g^{-ak} g^{ak} m = m.$$

5.3 Diffie-Hellman Schlüsselaustausch

Erstaunlicher Weise kann man einen geheimen Schlüssel über öffentliche Kanäle vereinbaren. Das Verfahren von Diffie und Hellman (mit dem das ElGamal-System in engem Zusammenhang steht) erlaubt dies unter Verwendung der diskreten Exponentialfunktion.
Alice und Bob wollen einen geheimen Schlüssel vereinbaren. Sie gehen nach Festlegung öffentlicher Parameter p, einer Primzahl, und $g \in \mathbb{Z}_p^*$ wie folgt vor:

1. Alice wählt eine geheime Zahl a und Bob eine geheime Zahl b.
2. Alice bildet $\alpha := g^a$ und sendet diesen Wert an Bob
   und Bob bildet $\beta := g^b$ und sendet diesen Wert an Alice.
3. Alice bildet β^a und Bob α^b. Wegen $\beta^a = (g^b)a = g^{ba} = g^{ab} = (g^a)^b = \alpha^b$ haben dann beide den gleichen Schlüssel.

Sind die Werte $\log_g \alpha$ und $\log_g \beta$ des diskreten Logarithmus nicht berechenbar, so ist der erzeugte Schlüssel sicher.

5.4 Elliptische Kurven

Als Gruppe G wird, wie schon erwähnt, oft $G = \mathbb{Z}_p^*$ oder allgemeiner $G = \mathrm{GF}(q)^*$ mit Primzahlpotenz q gewählt. Aber das in diesem Paragraphen erwähnte Verschlüsselungs- und auch die daraus abgeleiteten Signaturverfahren lassen sich auch mit anderen Gruppen realisieren. Für die Sicherheit des Systems ist es dabei wieder wichtig, dass die Berechnung des diskreten Logarithmus äußerst schwierig ist.
Für die Anwendungen wichtige Darstellungen von Gruppen stehen im Zusammenhang mit elliptischen Kurven (**e**lliptic **c**urves); die Public-key-Verfahren, die solche Kurven (in der sogenannten EC-Kryptographie, ECC) benutzen, liefern Alternativen zum RSA-Verfahren; sie kommen mit Primzahlen wesentlich geringerer Länge als RSA aus und können damit hardware-mäßig (in Smart Cards ohne Koprozessoren) billiger implementiert werden. Laut Cryptool (2003) ist eine gute Elliptische Kurve mit dem Parameter p einer Bitlänge von über 200 Bit [1] genau so sicher wie ein RSA–Modul von über 1024 Bit Länge, zumindestens nach dem gegenwärtigen Forschungsstand. Inzwischen wird vom Bundesamt für Sicherheit in der Informationstechnik (BSI) befristet bis Ende 2010 schon eine Schlüssellänge von 2048 Bit für RSA empfohlen.
Anmerkung am 31.7.2012: Die Ellipische Kurven Kryptographie wird auch zur Verschlüsselung beim (umstrittenen) Open-Source-Projekt "Bitcoin" für eine Digitalwährung verwendet. Vgl. Jürgen Müller: Bitcoins - Geld selbst drucken. LogIn Nr.171 (2011/2012), p. 31-36.

Zunächst führen wir kurz in die Theorie der elliptischen Kurven ein. (Diese lassen sich durch sogenannte elliptische Funktionen parametrisieren, daher der Name.)

5.4.1 Definitionen

Wir betrachten einen der Körper $K = \mathrm{GF}(p)$ für p prim[2], $p \neq 2$, $K = \mathbb{R}$ oder $K = \mathbb{Q}$. Eine **elliptische Kurve** (eine ebene *kubische Kurve*) ist definiert als die Menge $\mathcal{C}$ der Lösungen $(x, y) \in K^2$ einer kubischen Gleichung in zwei Variablen x und y, z. B. die Kurve der Gleichung

$$y^2 = x^3 + bx + c \quad \text{mit} \quad b, c \in K.$$

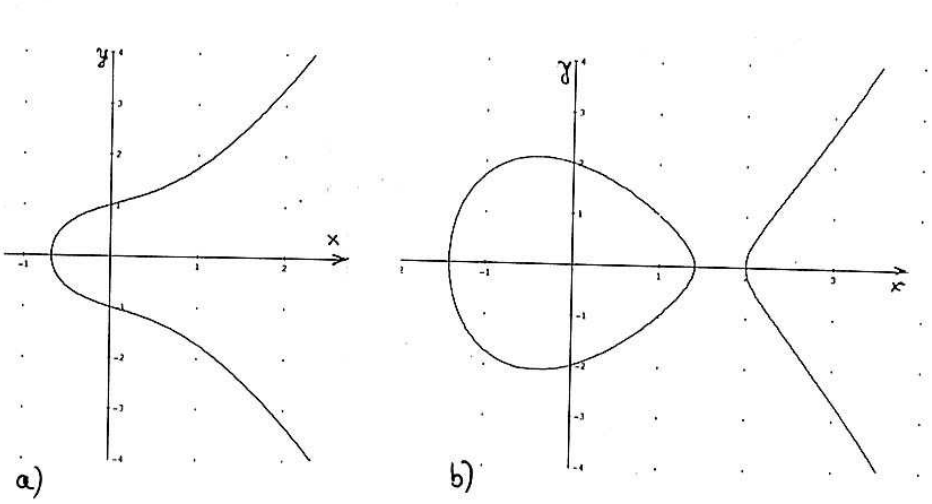
Die Kurve heißt dabei *nicht-singulär*, falls $D \neq 0$ für die "Diskriminante" $D = -4b^3 - 27c^2$ gilt. Beispiele mit $K = \mathbb{R}$ veranschaulicht Figur 5.1. Im reellen Fall heißt

$$\hat{\mathcal{C}} := \{(x, y) \in \mathcal{C} \mid x, y \in \mathbb{Q}\} \subseteq \mathcal{C}$$

die Menge der *rationalen Punkte* der Kurve $\mathcal{C}$. Auch die Punkte einer elliptischen Kurve über $\mathrm{GF}(p)$ mit p prim nennen wir "rational".

[1] nach anderen Quellen von 2^{139} für die Größenordnung des erzeugenden Punktes P aus einer elliptischen Kurve über $\mathrm{GF}(q)$

[2] Zur Behandlung des Falles $K = \mathrm{GF}(2^m)$ s.z.B. Cryptool!



Figur 5.2: Beispiele kubischer Kurven im Reellen
mit Gleichung $y^2 = f(x)$, wobei $f(x)$
a) eine reelle Nullstelle b) drei reelle Nullstellen besitzt

5.4.2 Beispiel einer elliptischen Kurve über GF(5)

Sei $\mathcal{C}$ die Kurve mit Gleichung $y^2 = x^3+x+1$ in der affinen Ebene über $K = GF(5)$. Sei $x \in K$; ist $f(x) := x^3 + x + 1$ ein Quadrat in K, so erhält man 'rationale' Punkte der Form (x, y) mit $x, y \in \mathrm{GF}(5)$ und $y^2 = f(x)$. Da $f(1) = 3$ kein Quadrat in GF(5) ist, folgt

$$\mathcal{C} = \{(0, \pm 1), (2, \pm 1), (3, \pm 1), (4, \pm 2)\}\,.$$

Meist nimmt man noch den gedachten "uneigentlichen Punkt" $\mathcal{O}$ in Richtung der y-Achse zu $\hat{\mathcal{C}}$ hinzu, durch den genau alle Geraden gehen sollen, die parallel zur y-Achse sind.

5.4.3 Addition von rationalen Punkten

Wir betrachten elliptische Kurven über $\mathbb{R}$ oder GF(p) für $p > 3$. Sei $\bar{\mathcal{C}}$ die Menge der rationalen Punkte der nicht-singulären Kurve der Gleichung

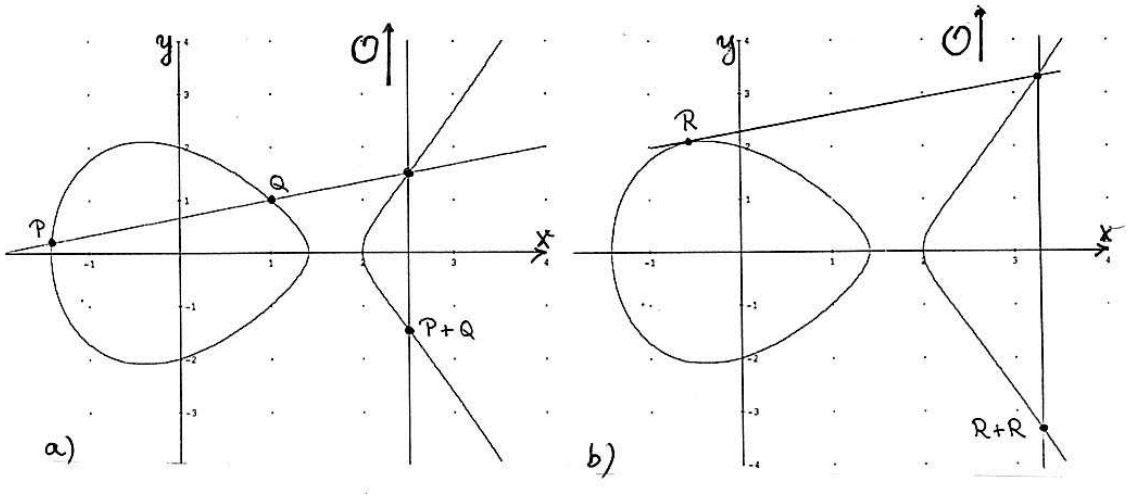
$$y^2 = x^3 + bx + c$$

einschließlich des uneigentlichen Punktes $\mathcal{O}$ in $y-$Richtung. Dann kann man auf $\bar{\mathcal{C}}$ eine Addition definieren. Wir beschreiben das Verfahren zunächst anschaulich geometrisch im reellen Fall, danach geben wir Formeln an.

a) Geometrische Beschreibung

Wir führen eine Verknüpfung + wie folgt ein: Sind P und Q mit $P \neq Q$ rationale Punkte, so spiegeln wir den dritten Schnittpunkt der Geraden

$g = PQ$ mit der Kurve an der $x-$Achse und definieren $P + Q$ als diesen Punkt. ist R rational, so sei $R + R$ der Punkt, den man durch Spiegelung an der $x-$Achse aus dem Schnittpunkt der Kurve mit der (lokalen) Tangenten in R an die Kurve (d.h. mit der Geraden, die R als Schnittpunkt doppelter Vielfachheit im algebraischen Sinne, also mit $f'(R) = 0$) besitzt (siehe Figur 5.2) erhält; man kann zeigen, dass $P + Q$ und $R + R$ existieren (und wieder rational sind).
Man kann zeigen, dass $(\overline{\mathcal{C}}, +)$ eine kommutative Gruppe ist. $\mathcal{O}$ ist das neutrale Element, das Kommutativgesetz ist klar, die Inversen erhält man wie in Figur 5.3 angedeutet. Der Beweis des Assoziativgesetzes ist etwas aufwändig.



Figur 5.3: Zur Verknüpfung rationaler Punkte von elliptischen Kurven

b) Algebraische Beschreibung
Jetzt kommen wir zur algebraischen Darstellung, die man durch Einsetzen der entsprechenden Geradengleichung in die Kurvengleichung erhält; alternativ kann man das folgende auch als Definition nehmen :

$$(x, y) + (x, -y) := \mathcal{O} \quad \text{und} \quad \mathcal{P} + \mathcal{O} := \mathcal{P} =: \mathcal{O} + \mathcal{P}$$

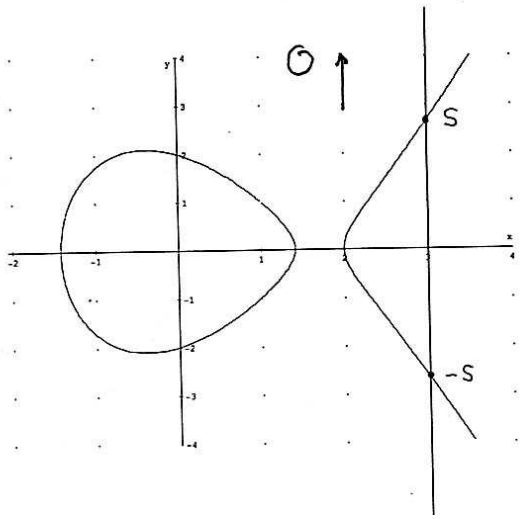
für alle Punkte (x, y) und P aus $\mathcal{C}$. Sind $P, Q \in \mathcal{C}$ mit $Q \neq -P$, und gilt $P = (p_1, p_2), Q = (q_1, q_2)$, so setzt man $P + Q := (s_1, s_2)$ mit

$$s_1 = \lambda^2 - p_1 - q_1 \quad \text{und} \quad s_2 = \lambda p_1 - p_2 - \lambda s_1$$

für

$$\lambda = \begin{cases} \frac{q_2 - p_2}{q_1 - p_1} & \text{falls} \quad P \neq Q, -Q \\ \frac{3p_1^2 + b}{2p_2} & \text{falls} \quad P = Q \ . \end{cases}$$

(Hierbei ist $y = \lambda x + (p_2 - \lambda p_1)$ die Gleichung der Geraden PQ.)



Figur 5.4: Zur Inversenbildung

5.4.4 Beispiel (Fortsetzung)

Im Fall der Kurve der Gleichung $y^2 = x^3 + x + 1$ über $K = GF(5)$ (s.o.) erhält man aus $\bar{\mathcal{C}}$ (einschließlich des uneigentlichen Punktes $\mathcal{O}$) eine abelsche Gruppe der Ordnung (d.h. Elementeanzahl) 9. Ist diese zyklisch (von einem Element erzeugt) oder elementar abelsch (jedes Element $\neq \mathcal{O}$ mit Ordnung 3)?
Für $P = (0, 1)$ ergibt sich aus den zitierten Formeln (mit $\lambda = \frac{3 \cdot 0 + 1}{2 \cdot 1} \equiv 3 (\mathrm{mod}\ 5)$):

$$P + P = 2P = (4, 2) \neq -P$$

sowie (mit $\lambda = \frac{1-2}{-4} \equiv 4 (\mathrm{mod}\ 5)$)

$$2P + P = 3P = (2, 1) \neq \mathcal{O}$$

und $4P = (3, -1)$; damit erzeugt P eine Untergruppe mit mehr als 3 Punkten; $(\mathcal{C} \cup \{\mathcal{O}\}, +)$ ist also eine zyklische Gruppe der Ordnung 9.
Die Werte der Logarithmusfunktion sind in folgender Tabelle (mit $Q = aP$) angegeben:

Q	$(0,1)$	$(0,4)$	$(2,1)$	$(2,4)$	$(3,1)$	$(3,4)$	$(4,2)$	$(4,3)$	$\mathcal{O}$
$a = \log_P Q$	1	8	3	6	5	4	2	7	0

5.4.5 Anmerkungen

1. Bei der Anwendung der Gruppe der rationalen Punkte einer elliptischer Kurve in der Kryptologie ist nicht so sehr die abstrakte Gruppe selbst, sondern ihre Darstellung von Bedeutung. Im Gegensatz zur ebenfalls zyklischen

Gruppe $(\mathbb{Z}_p, +)$, in der die Gleichung $xg = m$ durch Division zu lösen ist $(x = g/m)$, entspricht die Lösung der entsprechenden Gleichung über $(\bar{\mathcal{C}}, +)$ dem Auffinden des diskreten Logarithmus und ist daher schwerer zu finden.
2. H. W. Lenstra hat 1987 einen **Algorithmus zur Faktorisierung von ganzen Zahlen** vorgeschlagen, der elliptische Kurven benutzt, insbesondere die Addition auf deren Punkten. Bis zu welcher Größe man natürliche Zahlen faktorisieren kann, ist eine wichtige Frage für die Kryptoanalyse von öffentlichen Chiffriersystemen, z. B. des RSA-Systems.

Literatur zu §5.

Buchmann, J.: *Einführung in die Kryptographie.* Springer V., 2001[2].
Bundesamt für Sicherheit in der Informationstechnik(BSI):
http://www.bsi.bund.de/gshb/deutsch/m/m02164.html
Churchhouse, R.F.: *Codes and ciphers. Julius Caesar, the Enigma and the Internet.* Cambridge Univ.Press 2002.
Cohen, H.: *Zahlentheoretische Apekte der Kryptographie.* Informatik–Spektrum, 24. Juni 2001, p. 129-139.
CrypTool : Skript Mathematik und Kryptographie, Frankfurt/M. 2003
http:/www.CrypTool.de
Moreno, C.J.: *Curves over Finite Fields.* Cambridge, 1991.
Silverman, J. H. & Tate, J.: *Rational Points on elliptic Curves.* Springer V. , New York, 1992.
Werner, Annette: *Elliptische Kurven in der Kryptographie.* Springer V., Berlin, Heidelberg, 2002.

6 Quantenkryptographie

(frei nach S. Singh: Geheime Botschaften, Carl Hanser Verlag München/ Wien 2000)

6.1 Physikalische Grundlage (vereinfacht):

Durch einen vertikalen Polarisationsfilter werden

- vertikal polarisierte Photonen durchgelassen
- horizontal polarisierte Photonen blockiert
- diagonal polarisierte Photonen zur Hälfte blockiert und zur Hälfte durchgelassen und vertikal polarisiert.

6.2 Das Verfahren:

(1) Alice übermittelt an Bob zunächst eine Zufallsfolge aus 'Nullen' und 'Einsen', die sie ebenfalls zufällig rektilinear (horizontal und vertikal) oder diagonal nach dem untenstehenden Schema polarisiert.

Bob misst die Polarisation dieser Photonen. Da er nicht weiß, welches Polarisationsschema Alice für das jeweilige Photon verwandt hat, wechselt er zufällig zwischen dem + - Detektor und dem ×-Detektor (s. Figur 6.1 !).

Alices Schema	rektilinear				diagonal			
Alices Bit	1		0		1		0	
gesendetes Photon	↕		↔		↗		↘	
Bobs Detektor	+	×	+	×	+	×	+	×
Bobs Messung	↕	?	↔	?	?	↗	?	↘
Bobs Bit nach Kontakt mit Alice	1	-	0	-	-	1	-	0

Figur 6.1: Zur Quantenkryptographie.
('?' steht hier für einen unbestimmten Ausgang)

(2) Alice teilt Bob mit, bei welchen der Photonen er die richtige Messmethode verwendet hat, aber nicht, wie das Messergebnis lautet.

(3) Alice und Bob streichen die Messergebnisse, die Bob mit der falschen Methode erlangt hat; so erhalten sie ein Paar übereinstimmender Bit–Folgen.

(4) Alice und Bob kontrollieren die Übereinstimmung ihrer One time pads, indem sie eine Reihe von Stellen abfragen (und diese dann streichen).

(5) Wenn die Kontrolle Fehler ergibt, so nehmen sie an, dass die Polarisation der Photonen von einem Lauscher gemessen und dadurch in einigen Fällen die Polarisation geändert wurde. Sie beginnen dann erneut mit der Übertragung. Andernfalls haben sie – so die Theorie – mit fast absoluter Sicherheit ein geheimes Paar identischer One time pads.

E-mail-Adresse des Autors:
schulz@math.fu-berlin.de