

Mathematische Grundlagen von öffentlichen Verschlüsselungsverfahren

Ralph–Hardo Schulz

Lehrerfortbildungs-Veranstaltung
“Computersicherheit durch moderne Verfahren der
Kryptologie”

Es handelt sich hier um den 2. Teil einer Lehreinheit. Zuvor wurde in symmetrische Verfahren und in RSA als öffentliches Verfahren eingeführt.

1 Regeln für modulares Rechnen

Vorerfahrung: z.B. Uhrzeit 15 Uhr $\xrightarrow{\text{mod } 12}$ 3 Uhr (Nachmittag)
5 Stunden nach 21 Uhr $\xrightarrow{\text{mod } 24}$ 2 Uhr

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

1.1 Kongruenz modulo n

Es bezeichne $R_n(m)$ den Rest zwischen 0 und $n-1$ von m bei Division durch n . Andere Bezeichnungen: $m \bmod n$, $m \% n$. Wir definieren:

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

Wir vermerken, dass u.a. $R_n(a) \equiv a \pmod{n}$ folgt.

Es gelten folgende

1.2 Rechenregeln

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

Division einer Kongruenz durch t ist nur erlaubt, wenn t und n teilerfremd sind (vgl. Bsp. 1.3 c):

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

1.3 Beispiel für das Rechnen mit Resten:

a) $R_{12}(15) + R_{12}(11) = 3 + 11 = 14 \equiv 2 \pmod{12}$
 $\neq 2$

Man beachte also: $R_n(a_1 + a_2) = \mathbf{R}_n(R_n(a_1) + R_n(a_2))$.

b) Definiert man die Verknüpfung \oplus_n und \odot_n durch

$$a \oplus_n b := R_n(a + b) \quad \text{und} \quad a \odot_n b := R_n(a \cdot n),$$

so ergeben sich für $n = 5$ folgende Verknüpfungstafeln:

\oplus_5		0	1	2	3	4
0		0	1	2	3	4
1		1	2	3	4	0
2		2	3	4	0	1
3		3	4	0	1	2
4		4	0	1	2	3

\odot_5		0	1	2	3	4
0		0	0	0	0	0
1		0	1	2	3	4
2		0	2	4	1	3
3		0	3	1	4	2
4		0	4	3	2	1

c) Beispiele von Produktbildungen (bei Modul 10) mit und ohne Umkehrbarkeit:

a		0	1	2	3	4	5	6	7	8	9
$3 \cdot a \pmod{10}$		0	3	6	9	2	5	8	1	4	7

Die Umkehrung (Division durch 3) ist möglich, da $\text{ggT}(3, 10) = 1$ gilt.

a		0	1	2	3	4	5	6	7	8	9
$4 \cdot a \pmod{10}$		0	4	8	2	6	0	4	8	2	6

Division durch 4 ist nicht möglich, da $\text{ggT}(4, 10) \neq 1$ gilt.

1.4 Restklassen:

Die Kongruenz mod n ist eine Äquivalenzrelation. Jede Äquivalenzklasse (*Restklasse*) ist von der Form $\bar{a} = \{a + kn \mid k \in \mathbb{Z}\} = a + n\mathbb{Z}$, 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:

$$\mathbb{Z} = \bar{0} \cup \bar{1} \quad \text{für} \quad \bar{0} = 2\mathbb{Z} \quad \text{und} \quad \bar{1} = \{\dots, -7, -5, -3, -1, 1, 3, 5, 7, \dots\}$$

Die Restklassen modulo n fassen wir zusammen in der Menge

$$\mathbb{Z}_n := \{\bar{0}, \bar{1}, \dots, \overline{n-1}\}.$$

Jedes Element \bar{a} dieser Menge wird repräsentiert durch den Rest $R_m(a)$, der oft mit \bar{a} identifiziert wird.

Verknüpfungen auf \mathbb{Z}_n erhält man durch die Definitionen

$$\bar{a} + \bar{b} := \overline{a + b} \quad \text{und} \quad \bar{a} \cdot \bar{b} := \overline{a \cdot b};$$

diese Addition und diese Multiplikation entsprechen bei dem Übergang zu den Resten den Verknüpfungen \oplus_n und \odot_n .

Identifiziert man z.B. bei \mathbb{Z}_5 die Restklasse \bar{a} mit den Resten a (für $a \in \{0, 1, \dots, 4\}$), so erhält man die beiden Verknüpfungstafeln von 1.3 c).

Anmerkung: $(\mathbb{Z}_5, +, \cdot)$ ist dabei ein endlicher Körper (mit 5 Elementen), auch GF(5) genannt. Analog erhält man mit $(\mathbb{Z}_p, +, \cdot)$ für eine Primzahl p einen Körper mit p Elementen, als GF(p) bezeichnet.

2 Bestimmung der modularen Inversen

2.1 Teilen mit Rest

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

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

Hierbei ist $r = R_b(a)$ (in anderer Schreibweise: $a \bmod b$ oder $r = a \% b$).

Anmerkung 1: Es gilt: $\text{ggT}(a, b) = \text{ggT}(b, r) = \text{ggT}(b, a \bmod b)$.

Beweisskizze zur Anmerkung 1:

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

Anmerkung 2: Geometrisch ist die Division mit Rest durch das mehrfache Abtragen einer (kleineren) Strecke der Länge b auf einer (größeren) der Länge a zu veranschaulichen, s. Abbildung 1.

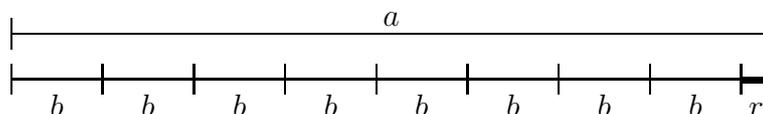


Abbildung 1: Geometrische Interpretation der Division mit Rest: $a = 8b + r$.

8

2.2 Euklidischer Algorithmus

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

Ursprung: Geometrische Kommensurabilitäts-Betrachtungen, schon bei Euklid: Die Strecken der Länge a und b lassen sich mit einem Maßstab der Länge $\text{ggT}(a, b)$ ausmessen.

Algorithmus:

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

$a_{\text{neu}} = a_{\text{alt}} \% a_{\text{mitte}}$

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

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

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

□

Beispiel:

Gesucht ist $\text{ggT}(48, 18)$.

$$\begin{aligned}
 (1) \quad & \underline{48} = 2 \cdot \underline{18} + \underline{12} \\
 (2) \quad & \underline{18} = 1 \cdot \underline{12} + \underline{6} \\
 (3) \quad & \underline{12} = 2 \cdot \underline{6} + \underline{0} \\
 (4) \quad & \underline{6}
 \end{aligned}$$

Also gilt $\text{ggT}(48, 18) = 6$.

Geometrische Interpretation siehe Abbildung 2 !

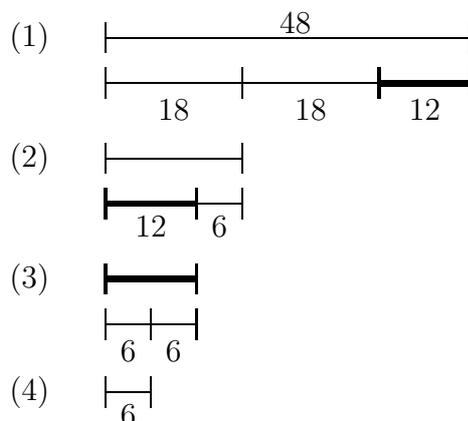


Abbildung 2: Geometrische Interpretation zum Euklidischen Algorithmus

2.3 Erweiterter Euklidischer Algorithmus

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

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

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.

Rekursion (s. auch Abbildung 3):

Start: $a = 1 \cdot a + 0 \cdot b$ und $b = 0 \cdot a + 1 \cdot b$.

Sei $a_{\text{alt}} = q \cdot a_{\text{mitte}} + a_{\text{neu}} = x_{\text{alt}}a + y_{\text{alt}}b$ und $a_{\text{mitte}} = x_{\text{mitte}}a + y_{\text{mitte}}b$!
Dann folgt

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

Daher setzt man:

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

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

Abbildung 3: Erweiterter Euklidischer Algorithmus, **Algorithmus von Berlekamp** zur Berechnung von $\text{ggT}(a, b)$ und von ganzen Zahlen x, y mit $\text{ggT}(a, b) = x \cdot a + y \cdot b$. (Python-Programm nach H. Witten 2001).

Beispiel:

$$\begin{array}{rcl}
60 & = & 2 \cdot 23 + 14 \quad x_3 = 1 - 2 \cdot 0 = 1 \quad y_3 = 0 - 2 \cdot 1 = -2 \\
& & 14 = x_3 \cdot 60 + y_3 \cdot 23 \\
& & = 1 \cdot 60 - 2 \cdot 23 \\
23 & = & 1 \cdot 14 + 9 \quad x_4 = 0 - 1 \cdot 1 = -1 \quad y_4 = 1 - 1 \cdot (-2) = 3 \\
& & 9 = x_4 \cdot 60 + y_4 \cdot 23 \\
& & = 1 \cdot 60 + 3 \cdot 23
\end{array}$$

usw. ...

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

2.4 Berechnung der Inversen

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

$$(*) \quad x \cdot a \equiv 1 \pmod{n},$$

existiert, falls $\text{ggT}(a, n) = 1$ ist. In letzterem Falle gibt es nach Abschnitt (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 $R_n(x_1) \cdot a + 0 \equiv 1 \pmod{n}$; also ist $x = R_n(x_1)$ modulo n invers zu a .

Algorithmus: s. Abbildung 4

```

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

```

Abbildung 4: Python-Programm zur Berechnung von modularen Inversen
(nach H. Witten 2001)

2.5 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 $\text{ggT}(e, (p-1)(q-1)) = 1$, so existieren nach dem Vielfachsummensatz (s. (2.3)) Zahlen x und y aus \mathbb{Z} mit

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

Diese lassen sich mit dem Erweiterten Euklidischen Algorithmus, s.(2.3), berechnen. Modulo $(p-1)(q-1)$ geht diese Gleichung (wie in (2.4)), hier für spezielles n) über in

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

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 $(p-1)(q-1) = 60 = a$; dann ist $b = e = 23$ teilerfremd und kann als öffentlicher Schlüssel gewählt werden. Wegen $1 = 5 \cdot 60 + (-13) \cdot 23$ folgt

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

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

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 $1, 2, 3, \dots, p-1$. Die Zahlen $1 \cdot a, 2 \cdot a, 3 \cdot a, \dots, (p-1) \cdot a$ sind ebenfalls $p-1$ verschiedene Zahlen; je zwei von ihnen haben verschiedene Reste (ungleich 0) \pmod{p} , da wegen $\text{ggT}(a, p) = 1$ gilt:

$$k \cdot a \equiv l \cdot a \pmod{p} \implies k \equiv l \pmod{p},$$

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 $\{1, 2, 3, \dots, p-1\}$, und es gilt

$$\{\overline{1 \cdot a}, \overline{2 \cdot a}, \overline{3 \cdot a}, \dots, \overline{(p-1) \cdot a}\} = \{\bar{1}, \bar{2}, \bar{3}, \dots, \overline{(p-1)}\}$$

und damit

$$\overline{1 \cdot 2 \cdot 3 \cdot \dots \cdot (p-1)} = \overline{1 \cdot a \cdot 2 \cdot a \cdot 3 \cdot a \cdot \dots \cdot (p-1) \cdot a} \quad \text{d.h. } ^1$$

$$1 \cdot 2 \cdot 3 \cdot \dots \cdot (p-1) \equiv [1 \cdot 2 \cdot 3 \cdot \dots \cdot (p-1)] \cdot a^{p-1} \pmod{p};$$

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 φ die Eulersche Phi-Funktion, also φ mit $\varphi(n)$ als der Anzahl der zu n teilerfremden Zahlen aus der Menge $\{1, \dots, n-1\}$; 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 $n = p \cdot q$ für zwei Primzahlen p, q

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

Insbesondere existiert ein t mit $de = t(p-1)(q-1) + 1$.

Für Zahlen m kleiner n (als Nachrichten) wird die Chiffrierung gemäß Abbildung 5 vorgenommen. Wir zeigen, 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 $c^d \equiv m^{ed} \pmod{n}$.

¹Das heißt für die Multiplikationstafel von \mathbb{Z}_p^* , dass das Produkt der Elemente der ersten Zeile und das der Elemente der a -ten Zeile übereinstimmen; also sind die Zeilenprodukte konstant. Im Beispiel von \mathbb{Z}_5^* aus 1.3 erhält man

\odot_5	1	2	3	4	Zeilenprodukt
1	1	2	3	4	$1 \odot 2 \odot 3 \odot 4 = 4$
2	2	4	1	3	$2 \odot 4 \odot 1 \odot 3 = 4$
3	3	1	4	2	$3 \odot 1 \odot 4 \odot 2 = 4$
4	4	3	2	1	$4 \odot 3 \odot 2 \odot 1 = 4$

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 $R_n(c^d) = m$. \square

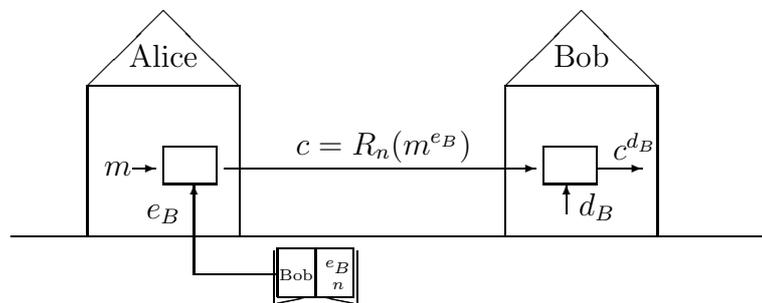


Abbildung 5: Schema der Chiffrierung und Dechiffrierung beim RSA-System

3.4 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.

3.4.1 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

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

Ist n eine zu testende Zahl, und findet man eine Zahl $a < n$ mit $a^{n-1} \not\equiv 1 \pmod{n}$, so folgt daraus, dass n keine Primzahl ist. Wenn hingegen $a^{n-1} \equiv 1 \pmod{n}$ 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. die DIFF-Studienbriefe [NF] oder Buchmann [Bu], s. die Literaturliste am Ende des Abschnitts!).

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.

3.4.2 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:

Miller-Rabin-Test:
 $a^m \equiv 1 \pmod{n}$ oder $a^{2^j m} \equiv -1 \pmod{n}$ für ein $j \in \{0, 1, \dots, s-1\}$.

Es gilt der folgende

Satz: *Jede Primzahl besteht den Miller-Rabin-Test.*

Beweis: Sei $p = 2^s \cdot m + 1$ Primzahl und $a^m \not\equiv 1 \pmod{p}$. Nach dem kleinen Satz von Fermat, s. (3.1), gilt aber zumindest $a^{2^s m} \equiv 1 \pmod{p}$. In der endlichen Folge $(a^m, a^{2m}, a^{2^2 m}, \dots, a^{2^s m})$ ist das letzte Glied also kongruent 1, eventuell aber schon ein vorangehendes Glied (und jedes nachfolgende), nicht aber das erste. Sei also

$$x^2 = a^{2^i m} \equiv 1 \pmod{p}$$

und i minimal mit dieser Eigenschaft; es ist dann 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 $x \not\equiv 1 \pmod{p}$ und folglich $a^{2^{i-1} m} = x \equiv -1 \pmod{p}$. \square

Beispiel: Die Carmichael Zahl $n = 561$ besteht wegen $2^{35} \equiv 263 \pmod{561}$ und $2^{140} \equiv 67 \pmod{561}$ sowie $2^{280} \equiv 1 \pmod{561}$ nicht den Miller-Rabin-Test.

Anmerkungen:

1. Der Miller-Rabin-Test ist strenger als der Fermat-Test: Aus $a^m \equiv 1 \pmod{m}$ folgt $a^{n-1} = (a^m)^{2^s} \equiv 1 \pmod{n}$; aus $a^{2^j m} \equiv -1 \pmod{n}$ für $j < s$ ergibt sich

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

Jede Zahl, die den Miller-Rabin-Test besteht, passiert also auch den Fermat-Test erfolgreich.

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 zusammengesetzte Zahl als solche erkennen.

3.5 Faktorisierung

3.5.1 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-Studienbriefe [NF] !).

3.5.2 Quadratisches Sieb

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

$$x^2 \equiv y^2 \pmod{n} \quad \text{und} \quad x \not\equiv \pm y \pmod{n}$$

bestimmt werden. (Es gilt dann: n teilt $x^2 - y^2$, aber nicht $x - y$ und $x + y$, sodass man mit $\text{ggT}(x - y, n)$ einen echten Teiler von n gefunden hat. Weiteres s. z.B. Buchmann [Bu] !).

3.5.3 Anmerkungen zur Faktorisierung

1. Auch diese und andere (zum Teil probabilistische) Faktorisierungsalgorithmen 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 534 binären Stellen. Im Jahr 2003 konnten Franke et al. sogar die Zahl “RSA-576” mit 576 binären und 174 dezimalen Stellen faktorisieren, allerdings mit knapp 3 Monaten Rechenzeit auf mehr als 100 Workstations (s.z.B. [RSA]). Bisher ist es hingegen nicht gelungen, die Zahl “RSA-640” (mit 193 Dezimalstellen) in Faktoren zu zerlegen.
2. Vom Bundesamt für Sicherheit in der Informationstechnik [BSI] wird für das RSA-Verfahren befristet bis Ende 2007 eine Mindest-Schlüssellänge von 1024 Bits, bis Ende 2010 von 1984 Bits gefordert, und schon jetzt bis Ende 2010 eine Schlüssellänge von 2048 Bit empfohlen.² Dabei sollten die beiden Primzahlen p und q mit $n = p \cdot q$ von der gleichen Größenordnung sein.
3. Bei ungünstiger Wahl von p und q ist eine Faktorisierung leichter möglich, z.B. wenn p und q zu nahe beieinander liegen (“Kollision”), sodass sie in der Nähe von \sqrt{n} gesucht und gefunden werden können.
4. 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

²Laut Netscape ist die Ausfuhr von Software mit einem RSA-Modul von über 512 Bits aus den USA nicht erlaubt.

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! (S. auch [Schu96] !)

Literaturauswahl zu §1 bis §3:

- [**Bu**] BUCHMANN, J.: Einführung in die Kryptographie. Springer Verlag, Wiesbaden, Heidelberg, 1999, 2001².
- [**Cr**] CRYPTTOOL-Skript: Mathematik und Kryptographie.
<http://www.CrypTool.de>
- [**NF**] NIEDERDRENK-FELGNER: Algorithmen der elementaren Zahlentheorie. DIFF – Studienbriefe CM1, Tübingen 1988.
- [**Ba**] BARTHOLOMÉ et al.: Zahlentheorie für Einsteiger, Vieweg Verlag, Braunschweig 1996.
- [**BNS**] BEUTELSPACHER, A., NEUMANN, H. und SCHWARZPAUL, T.: Kryptografie in Theorie und Praxis. Vieweg Verlag, Wiesbaden. 2005.
- [**Er**] ERTEL, W.: Angewandte Kryptographie. Fachbuchverlag Leipzig 2003².
- [**Sch93**] SCHULZ, R.-H.: Primzahlen in öffentlichen Chiffrierverfahren. Mathematik lehren 61 (1993) 56-64.
- [**Sch96**] SCHULZ, R.-H.: Primfaktorzerlegung. Experimente zum Zeitaufwand. LogIn 16 (1996) 22-26.
- [**Sch03**] SCHULZ, R.-H.: Codierungstheorie. Eine Einführung. Kap.19. Vieweg Verlag, Wiesbaden 2003².
- [**BSI**] Bundesamt für Sicherheit in der Informationstechnik (BSI).
<http://www.bsi.bund.de/esig/basics/techbas/krypto/>
- [**RSA**] RSA Security. <http://www.rsasecurity.com/>
- [**Zi**] ZIEGENBALG, J.: Materialien und interaktive Simulationen zu Codierung und Kryptographie.
<http://www.ph-karlsruhe.de/~ziegenbalg/materialien-homepage-jzbg/cc-interaktiv/index.htm>

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.B. 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 (s. auch unten).

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 Tabelle 1 angegeben.

Tabelle 1: Signatur durch Alice mittels des RSA-Verfahrens
 Dabei bezeichnet (e_A, n) den öffentlichen und d_A den
 geheimen Schlüssel von A .

<p>Alice berechnet $y = x^{d_A} \pmod n$ und sendet (x, y) an Bob</p> <p>Bob berechnet $x' = y^{e_A} \pmod n$ und prüft “$x' = x$?”</p>
--

Anwendungsbeispiele 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 der Firma Montrada GmbH, in die die Firma Cyber Cash übergegangen ist, werden die Kreditkarten-Daten des (Internet-) Kunden durch eServer mit dem RSA-Verfahren verschlüsselt und elektronisch an den Händler gesandt, der seine Daten hinzufügt und an einen virtuellen POS (“POSH-Service”) und von dort an die Autorisierungsstelle weiterleitet. Diese entschlüsselt die Daten und kontrolliert bei dem Kreditkarteninstitut die Gültigkeit der Karte und die Deckung. Erst danach erfolgt die Abbuchung. (Das Verfahren wird von der Commerzbank und der Dresdner Bank angewandt).

4.3 Anonymes Digitales Geld

Das wichtigste Verfahren der ersten Generation hierbei, das **E-cash** der Firma DigiCash, beruht auf einem Modell von D.Chaum. (Es wurde u.a. von der Deutschen Bank unterstützt.)

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 trotz Zertifizierung zu garantieren, wird W , wie in Tabelle 2 und Abbildung 6 beschrieben, **blind signiert**.

Nach dem blinden Signieren durch die Bank besitzt der Kunde nun in W^d eine anonyme elektronische Münze, einen **Cyberdollar** (s. Abbildung 6). Die Münze M lässt sich durch Bildung von $M^e \pmod n \equiv W^{de} \pmod n \equiv W \pmod n$ 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

³Als mechanisches Modell für die blinde Signatur dient das Folgende: Durch eingelegetes Kohlepapier drückt sich die auf dem Couvert geleistete Unterschrift auf das nicht sichtbare Dokument durch.

Tabelle 2: Blinde Signatur mittels des RSA-Verfahrens.

<p>Blinde Signatur:</p> <ul style="list-style-type: none"> • <i>Ziel:</i> Die Bank soll W unterschreiben, ohne später W dem Kunden zuordnen zu können. • <i>“Blinding-Faktor”:</i> Der Kunde erzeugt eine Zufallszahl C, bildet C^e und sendet $S := C^e \cdot W \pmod{n}$ an die Bank. (Hierbei sei (e, n) ein öffentlicher und d der bei der Bank deponierte zugehörige geheime Schlüssel im RSA-System.) • <i>Blinde Unterschrift:</i> Die Bank zertifiziert S durch ihre Signatur zu $T = S^d \pmod{n}$ und sendet T an den Kunden. 3) • Der Kunde bildet $M := C^{-1} \cdot T \equiv C^{-1} S^d \equiv C^{-1} C^{ed} W^d \equiv W^d \pmod{n}$ <p>und hat damit das Ziel der blinden Signatur erreicht.</p>
--

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, \dots, k_r und erhält (evtl. nach sogenanntem Hashen des Textes) Geheimtexte $c_1 = f(k_1, m), \dots, c_r = f(k_r, m)$; die Werte k_i und c_i lässt er von der Bank blind zu $\text{sig}(k_i)$ bzw. $\text{sig}(c_i)$ signieren (s. Abbildung 2). Der Verkäufer wählt dann r Bits b_1, \dots, 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. Man unterscheidet zwei Modellgruppen:

1. Zur Verifizierung des Gutscheins wird ganz $G = (G_0, G_Z)$ übertragen.

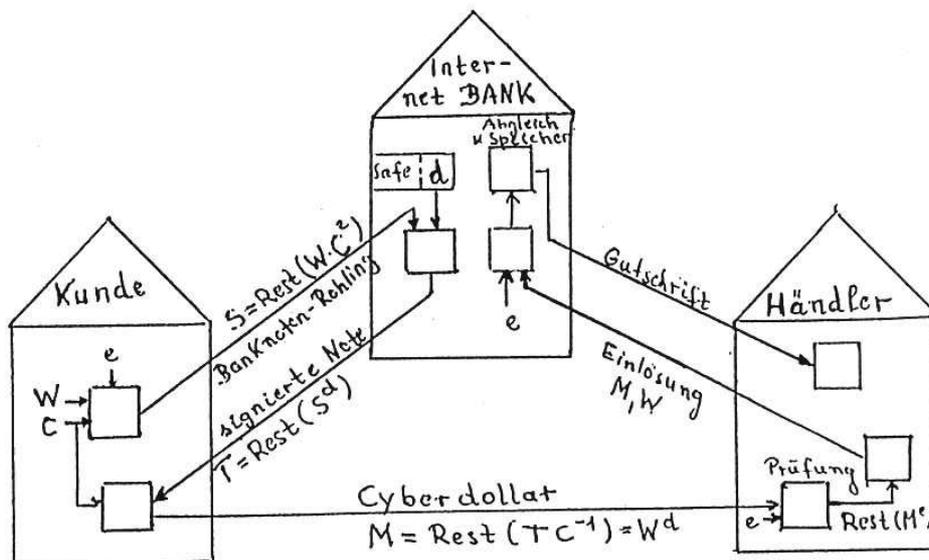


Abbildung 6: Zum E-Cash - Verfahren

Ist dabei

$$G_Z = f_K(G_0)$$

mit dem Schlüssel K gebildet worden, so lässt sich G_0 bestätigen durch Anwendung des Schlüssels K (im symmetrischen Fall) bzw. des zum geheimen Schlüssel gehörenden öffentlichen Schlüssels K' (im asymmetrischen Fall).

- Es wird nur G_0 übertragen (damit ist Duplizierung durch Abhören nicht möglich). Durch eine geheime Funktion F können die Partner den gleichen geheimen Schlüssel G_Z erhalten, der dann eine Verifizierung durch Anwendung auf eine von der Akzeptanzstelle versandte Zufallszahl oder durch ein "zero knowledge" Protokoll erlaubt.

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.
Beispiel: Telefonieren in einem Mobiltelefonnetz mit Abbuchung der Kosten vom Bank-Konto.
- Um Wiedereinspielung abgefangener Nachrichten zu verhindern, sind bei vielen Homebanking-Vorgängen vorher ausgegebene **Transaktionsnummern** "TAN"s in Verwendung.

Literatur zu §4:

- [**Bau**] BAUMANN, R. : Digitales Geld. Bestellen und Bezahlen im Internet. LOG IN **17/2** (1997) 30-38.
- [**BTP**] BEUTELSPACHER, A. , T.HUESKE und A.PFAU: Kann man mit Bits bezahlen? Informatik-Spektrum **16** (1993) 99-106.
- [**Er**] ERTEL, W.: Angewandte Kryptographie. Fachbuchverlag Leipzig 2003².
- [**We**] WEIS, RÜDIGER: Zahlen Sie bar oder digital? Protokolle für sicheres Geld. PC Magazin Spezial 5.98 (Kryptographie und Netzsicherheit) 1998, p. 94-97.
- [**Schm**] SCHMEH, K.: Die Welt der geheimen Zeichen. W3L-Verlag. Herdecke/Dortmund, 2004 §3.13.
- [**Schn**] U.SCHNEPPE: Elektronischer Zahlungsverkehr im Internet. <http://ulrich.schneppe.bei.t-online.de/s1916/start.htm>

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) Gruppe (z.B. der multiplikativen Gruppe $\mathbb{Z}_p^* = \{1, \dots, p-1\}$) heranziehen, solange die Berechnung der Umkehrung praktisch unmöglich ist. Sei also G eine endliche Gruppe und $g \in G$. Wichtig ist dann die von g erzeugte zyklische Untergruppe

$$H = \langle g \rangle = \{g^i \mid i \geq 0\} = \{g, g^2, g^3, \dots, g^m\};$$

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 (Elementanzahl) der von g erzeugten Gruppe H .

Oft wählt man g speziell als erzeugendes Element von $G = \mathbb{Z}_p^*$.

Beispiel: $p = 11$:

Für $H = \langle 2 \rangle$ ergibt sich Tabelle 3 und $|G| = |H| = |\mathbb{Z}_{11}^*| = 11 - 1$.

Tabelle 3: Zur zyklischen Gruppe \mathbb{Z}_{11}^* .

2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
2	4	8	$16 \equiv 5$	10	$20 \equiv 9$	$18 \equiv 7$	$14 \equiv 3$	6	$12 \equiv 1$

Nicht jedes Element erzeugt \mathbb{Z}_{11}^* : $|\langle 4 \rangle| = 5 \neq 10$.

Als Einwegfunktion kommt die “diskrete Exponentialfunktion” in Frage:

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

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 (Fortsetzung): Tabelle 4 zeigt die Wertetabelle der Logarithmusfunktion im Fall $G = \mathbb{Z}_{11}^* = H$ und $g = 2$.

Tabelle 4: Wertetabelle der diskreten Logarithmusfunktion \log_2 auf \mathbb{Z}_{11}^*

$2^k \pmod{11}$	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 $\text{GF}(q)^*$ “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 und $g \in G$ derart, dass die Berechnung des diskreten Logarithmus \log_g in $H := \langle g \rangle$ fast unmöglich ist,
- a eine natürliche Zahl mit $1 < a < |H|$ und
- $h := g^a$.

Die Ver- bzw. Entschlüsselung erfolgt wie in Tabelle 5 angegeben.

Tabelle 5: Verallgemeinertes ElGamal-System

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 $k \in \{2, \dots, H - 1\}$
Entschlüsselung	$D(x, y) := (x^a)^{-1} \cdot y$

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

5.3 Babystep–Giantstep–Algorithmus

Aus den vielen Algorithmen zur Berechnung des Diskreten Logarithmus wählen wir den Babystep–Giantstep–Algorithmus von Shanks aus und beschreiben hier kurz die mathematischen Grundlagen.

5.3.1 Aufgabe, Heuristik und Beschreibung des Algorithmus

Aufgabenstellung: Wie in Abschnitt 5.2 seien G eine endliche Gruppe, $g \in G$ und $h \in H := \langle g \rangle$. Gesucht ist $a = \log_g h$.

Heuristik: Sei $|H| = o(g) =: n$; wir setzen $w := \lceil \sqrt{n} \rceil$; (wegen $\sqrt{n} \leq w$ ist damit $w^2 \geq n$).

Jede natürliche Zahl a mit $1 < a \leq n - 1$ lässt sich (durch Division mit Rest) in der folgenden Form darstellen:

$$a = j \cdot w + r \quad \text{mit } j, r \in \{0, \dots, w - 1\}.$$

Folglich gilt für $a \in \{2, \dots, n - 1\}$ auch $h = g^a = g^{j \cdot w + r}$, was zu

$$h \cdot g^{-r} = g^{j \cdot w}$$

äquivalent ist. (Alle Potenzen g^a sind auch erfasst.)

Beschreibung: Bei der Bestimmung von a sind r und j unbekannt. Von diesen beiden steht eines auf der linken und das andere auf der rechten Seite der oben hergeleiteten Gleichung $h \cdot g^{-r} = g^{j \cdot w}$; daher betrachtet man eine der Seiten zunächst gesondert.

Die linke Seite legt die Berechnung der *Babystep–Liste*

$$B = \{h \cdot g^{-r} \mid r = 0, \dots, w - 1\}$$

nahe; bei der Bestimmung der Elemente der *Giantstep–Liste*

$$\{1, g^w, \dots, (g^w)^{(w-1)}\}$$

vergleicht man in jedem Schritt, ob $(g^w)^j$ in B vorkommt. In diesem Fall, also für $(g^w)^j = h \cdot g^{-r}$ ist $h = g^{wj+r}$, also $\log_g a$ mittels j und r als $a = wj + r$ bestimmt.

(Die Effizienz des viel Speicher verbrauchenden Verfahrens lässt sich durch Anwendung einer Hash-Funktion auf die Einträge der Babystep-Liste verbessern.)

5.3.2 Komplexität des Algorithmus

Wieviel Gruppenoperationen und Vergleiche benötigt man bei diesem Algorithmus maximal? Für die Berechnung der Babystep-Liste sind eine Invertierung und w Multiplikationen nötig; für die Giantstep-Liste braucht man weniger als w Multiplikationen zur Berechnung von g^w und höchstens w Gruppenoperationen bis zum letzten Element der Liste. In manchen Fällen (z.B. bei elliptischen Kurven, s. Abschnitt 5.5) ist die Invertierung sehr einfach, sodass dann etwa $3w$ Gruppenoperationen nötig sind, also $O(\sqrt{|H|})$ Operationen. Weiter sind die Vergleiche (mittels Hashliste als konstant bewertet) und Speicherschritte zu berücksichtigen.

Wenn die auftretenden Zahlen eine Länge von etwa 1000 Bits haben, so braucht man bei der Berechnung der diskreten Exponentialfunktion, also von g^k , (mittels Square and Multiply) ca 2000 Multiplikationen, zur Berechnung des diskreten Logarithmus mittels des Babystep-Giantstep-Algorithmus aber ca $\sqrt{2^{1000}} = 2^{500} \approx 10^{150}$ Operationen (s. Cryptool). Dies ist Ausdruck dessen, dass für große Parameter die diskrete Exponentialfunktion als Einwegfunktion aufgefasst werden kann.

5.4 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 in Tabelle 6 vor:

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

Tabelle 6: Diffie-Hellman Schlüssel-Austausch

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.

5.5 Elliptische Kurven

Als Gruppe G wird, wie schon erwähnt, oft $G = \mathbb{Z}_p^*$ oder allgemeiner $G = \text{GF}(q)^*$ 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 (**elliptic curves**); 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 ⁴ genau so sicher wie ein RSA-Modul von über 1024 Bit Länge⁵ zumindestens nach dem gegenwärtigen Forschungsstand. Zunächst führen wir kurz in die Theorie der elliptischen Kurven ein. (Diese lassen sich durch sogenannte elliptische Funktionen parametrisieren, daher der Name.)

5.5.1 Definitionen

Wir betrachten als Zahlbereich $K = \text{GF}(p) = (\mathbb{Z}_p, +, \cdot)$ für p prim⁶, $p \neq 2$, $K = \mathbb{R}$ 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

⁴nach anderen Quellen von 2^{139} für die Größenordnung des erzeugenden Punktes P aus einer elliptischen Kurve über $\text{GF}(q)$

⁵Wie oben schon erwähnt, sind laut Bundesamt für Sicherheit in der Informationstechnik 1024 Bit die Mindestgröße für (zur Zeit als sicher angesehene) RSA-Schlüssel (bis Ende 2007); empfohlen sind aber 2048 Bit.

⁶Zur Behandlung des Falles $K = \text{GF}(2^m)$ s.z.B. Cryptool!

in zwei Variablen x und y , z. B. die Kurve der Gleichung

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

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 Abbildung 7. Im reellen Fall heißt

$$\hat{\mathcal{C}} := \mathcal{C} \cap \mathbb{Q}^2 \subseteq \mathcal{C}$$

die Menge der *rationalen Punkte* der Kurve \mathcal{C} . Auch die Punkte einer elliptischen Kurve über \mathbb{Z}_p mit p prim nennen wir ‘‘rational’’.

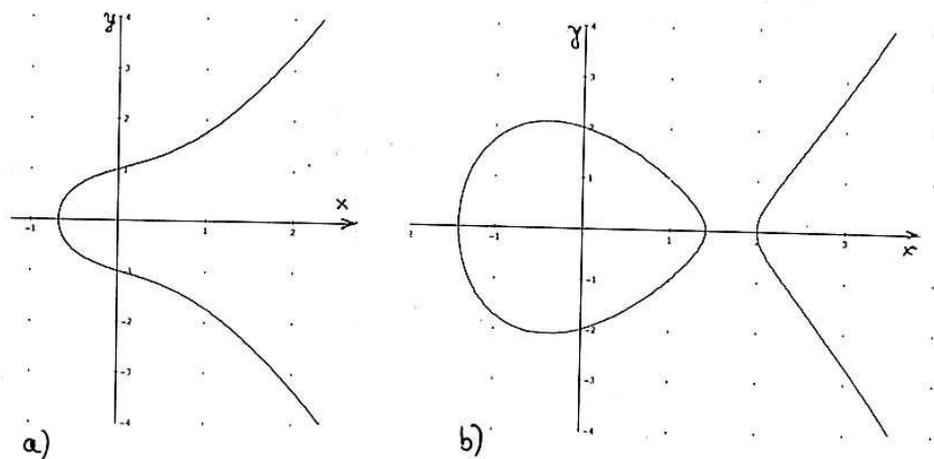


Abbildung 7: Beispiele kubischer Kurven im Reellen
mit Gleichung $y^2 = f(x)$, wobei $f(x)$

a) eine reelle Nullstelle b) drei reelle Nullstellen besitzt

5.5.2 Beispiel einer elliptischen Kurve über \mathbb{Z}_5

Sei \mathcal{C} die Kurve mit Gleichung $y^2 = x^3 + x + 1$ in der affinen Ebene über $K = \mathbb{Z}_5$. 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 $x, y \in \mathbb{Z}_5$ und $y^2 = f(x)$. Da $f(1) = 3$ kein Quadrat in \mathbb{Z}_5 ist, folgt

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

Meist nimmt man noch den uneigentlichen Punkt \mathcal{O} in Richtung der y -Achse zu $\hat{\mathcal{C}}$ hinzu.

5.5.3 Addition von rationalen Punkten

Wir betrachten elliptische Kurven über \mathbb{R} oder $\text{GF}(p)$ 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 Abbildung 8) erhält; man kann zeigen, dass $P + Q$ und $R + R$ existieren (und wieder rational sind).

Man kann zeigen, dass $(\bar{\mathcal{C}}, +)$ eine kommutative Gruppe ist. \mathcal{O} ist das neutrale Element, das Kommutativgesetz ist klar, die Inversen erhält man wie in Abbildung 9 angedeutet. Der Beweis des Assoziativgesetzes ist etwas aufwändig.

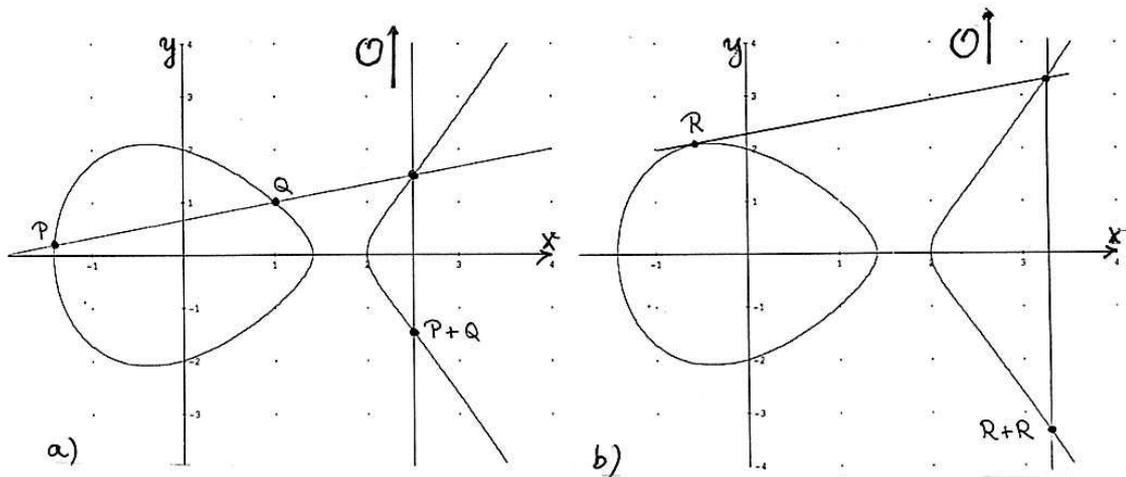
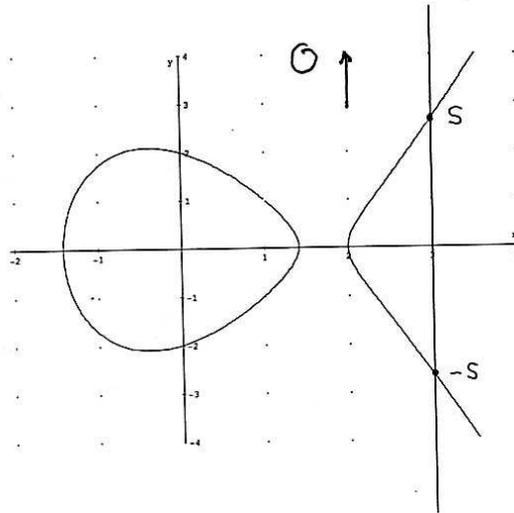


Abbildung 8: Zur Verknüpfung rationaler Punkte von elliptischen Kurven

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 :

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



Abbildungung 9: Zur Inversenbildung

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

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

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

5.5.4 Beispiel (Fortsetzung)

Im Fall der Kurve der Gleichung $y^2 = x^3 + x + 1$ über $K = \mathbb{Z}_5$ (s. Abschnitt 5.5.2) erhält man aus $\bar{\mathcal{C}}$ (einschließlich des uneigentlichen Punktes \mathcal{O}) eine Gruppe der Ordnung 9. Ist diese zyklisch (von einem Element erzeugt)? Für $P = (0, 1)$ ergibt sich aus den zitierten Formeln (mit $\lambda = \frac{3 \cdot 0 + 1}{2 \cdot 1} \equiv 3 \pmod{5}$):

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

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

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

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 Tabelle 7 (mit $Q = aP$) angegeben:

Tabelle 7: Die Logarithmusfunktion $\log_P Q$ des Beispiels 5.5.4

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

5.5.5 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 ($x = m/g$), entspricht die Lösung der entsprechenden Gleichung über $(\bar{\mathbb{C}}, +)$ dem Auffinden des diskreten Logarithmus und ist daher schwerer zu finden.
2. Ohne die verwendeten Begriffe definieren zu können, zitieren wir den

Satz von HASSE:

Ist \mathcal{C} eine nicht-singuläre irreduzible Kurve definiert über $GF(p)$, dann ist die Anzahl der Punkte auf \mathcal{C} mit Koordinaten in $GF(p)$ gleich $p + \varepsilon$ mit $|\varepsilon| \leq 2\sqrt{p}$.

Der Satz stellt damit sicher, dass über $GF(p)$ eine elliptische Kurve approximativ p Punkte (für großes p) enthält.

3. 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.

- [Bu] BUCHMANN, J.: *Einführung in die Kryptographie*. Springer V., 2001².
- [Ch] CHURCHHOUSE, R.F.: *Codes and ciphers. Julius Caesar, the Enigma and the Internet*. Cambridge Univ.Press 2002.
- [Co] COHEN, H.: *Zahlentheoretische Aspekte der Kryptographie*. Informatik-Spektrum, 24. Juni 2001, p. 129-139.
- [Cr] *CrypTool : Skript Mathematik und Kryptographie*, Frankfurt/M. 2003
<http://www.CrypTool.de>

- [Mo] MORENO, C.J.: *Curves over Finite Fields*. Cambridge, 1991.
- [ST] SILVERMAN, J. H. & TATE, J.: *Rational Points on elliptic Curves*. Springer V. , New York, 1992.
- [We] 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 Polarisationschema Alice für das jeweilige Photon verwandt hat, wechselt er zufällig zwischen dem $+$ -Detektor und dem \times -Detektor (s. Abbildung 10 !).

- (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 verändert wurde. Sie beginnen dann erneut mit der Übertragung.

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

Abbildung 10: Zur Quantenkryptographie.
 ('?' steht hier für einen unbestimmten Ausgang)

Andernfalls haben sie – so die Theorie – mit absoluter Sicherheit ein geheimes Paar identischer One time pads.

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

25. Februar 2005