Mathematische Grundlagen der Kryptographie

Skript zum Kurs an der Informatikabteilung der HSR
Sommersemester 2001

Dr. Andreas Müller

<andreas.mueller@othello.ch>

Mathematische Grundlagen der Kryptographie

Dr. Andreas Müller

INHALT

ABBILDUNGSVERZEICHNIS

Kapitel 1. Notation

In der Kryptographie werden Texte, also Folgen von Zeichen eines Alphabetes, mit einem Verschlüsselungsverfahren in einen Chiffrattext umgewandelt. In diesem Kapitel führen wir einige Notationen ein, welche uns in späteren Kapiteln von Nutzen sein werden.

1.1. Texte.

Ein Klar- oder Plaintext p ist eine Folge von Zeichen aus dem Klartextalphabet $\mathcal{P}$, die wir im Falle eines endlichen Textes der Länge n mit

$$p = (p_0, p_1, \ldots, p_{n-1}) \in \mathcal{P}^n$$

bezeichnen. Unendliche Klartextfolgen bezeichnen wir auch mit $(p_i)_{i\in\mathbb{N}} \in \mathcal{P}^{\mathbb{N}}$. In beiden Fällen sind alle $p_i \in \mathcal{P}$. Die Länge eines Textes kürzen wir mit $|p|$ ab.

Chiffrattexte werden mit der gleichen Notation beschrieben, das Alphabet des Chiffrates braucht nicht mit dem Alphabet des Klartextes übereinzustimmen. So verwendet ein englischer Text ausschliesslich die Zeichen mit Werten zwischen 33 und 127, doch die meisten Verschlüsselungsverfahren werden daraus ein Chiffrat erzeugen, in dem alle Bytes zwischen 0 und 255 mit gleicher Wahrscheinlichtkeit vorkommen.

Damit wir uns nicht weiter um die Länge von Texte kümmern müssen, führen wir die Notation $\hat{\mathcal{P}}$ für die Vereinigung der Mengen $\mathcal{P}^n$ ein.

1.1.1. Verkettung. Endliche Texte können verkettet werden, diese Operation wird mit Hilfe des Komma-Operators bezeichnet:

$$,: \hat{\mathcal{P}} \times \hat{\mathcal{P}} \to \hat{\mathcal{P}}: ((a_i)_{0\le i<n}, (b_j)_{0\le j<n}) \mapsto (a_0, \ldots, a_{n-1}, b_0, \ldots, b_{m-1}).$$

Die Länge der Verkettung zweier Texte ist die Summe der Länge der Einzeltexte, oder $|a, b| = |a| + |b|$.

1.1.2. Verknüpfungen. Falls auf dem Alphabet $\mathcal{P}$ ein Verknüfung g definiert sind, kann daraus eine Verknüpfung der Texte konstruiert werden. So gehört zur Abbildung

$$g: \mathcal{P} \times \mathcal{P} \to \mathcal{P}$$

die Abbildung

$$\hat{g}\colon \hat{\mathcal{P}} \times \hat{\mathcal{P}} \to \hat{\mathcal{P}}\colon ((a_i)_{0\le i<n}, (b_j)_{0\le j<m}) \mapsto (g(a_i, b_i))_{0\le i<\min(n,m)}.$$

Im Falle von Bytes, die bitweise mit XOR verknüpft werden können, notiert $\oplus$, entsteht eine ebenfalls mit $\oplus$ bezeichnete Operation auf Bytefolgen.

1.2. Verschlüsselung.

Ein Verschlüsselungsalgorithmus E ist eine invertierbare, von einem Schlüssel k abhängige Abbildung

$$E_k\colon \hat{\mathcal{P}} \to \hat{\mathcal{C}},$$

mit der Umkehrabbildung

$$D_k\colon \hat{\mathcal{C}} \to \hat{\mathcal{P}}.$$

In vielen Fällen lässt sich die Umkehrabbildung als $E_{k'}$ mit einem geeigneten Wert für k' realisieren.

Aus einer Abbildung $E_k\colon \mathcal{P} \to \mathcal{C}$ lässt sich auch eine Abbildung zwischen den Folgen bilden, indem einzelne Folgenelemente abgebildet werden:

$$E_k\colon \hat{\mathcal{P}} \to \hat{\mathcal{C}}\colon (p_i)_{0\le i<n} \mapsto (E_k(p_i))_{0\le i<n}.$$

Leistungsfähige Algorithmen verwenden jedoch meist eine Verkettung von Klartextzeichen, um daraus mehrere Chiffratzeichen zu konstruieren. Die Blockverschlüssler zum Beispiel fassen jeweils b Zeichen zu einem Block zusammen, um daraus b Zeichen Chiffrat zu berechnen. Der Verschlüssler ist also eine Abbildung

$$E\colon \mathcal{P}^b \to \mathcal{C}^b.$$

Der Klartext muss ein Vielfaches von b als Länge haben, also

$$p \in (\mathcal{P}^b)^l \cong \mathcal{P}^{bl}.$$

Die Verschlüsselung des gesammten Klartextes ist dann

$$E_k^l\colon (\mathcal{P}^b)^l \to (\mathcal{C}^b)^l\colon (b_0, \ldots, b_{l-1}) \mapsto (E_k(b_0), \ldots, E_k(b_{l-1})).$$

Kapitel 2. Klassische Verfahren

Die Kryptographie dürfte etwa so alt sein, wie sich kriegführende Parteien der Schrift bedienen. Die Verwendung kryptographischer Verfahren soll die übermittelte Information einem Gegner auch dann unzugänglich machen, wenn es ihm gelingen sollte, den Boten oder die Meldung abzugfangen. Das verwendete Verfahren ist stark genug, wenn die Mitteilung so lange vertraulich bleibt, bis sie für den Gegner taktisch wertlos geworden ist. Dieser Grundsatz gilt auch heute: die Möglichkeiten des Gegners und der bei Verlust der Vertraulichkeit der Mitteilungen eintretende Schaden bestimmen, welche Sicherheit die Verschlüsselung bieten muss.

2.1. Einfache Bespiele.

In diesem Abschnitt werden einfache Verfahren betrachtet, die von geschichtlicher Bedeutung waren, aber mit heutigen Mitteln innert Sekundenbruchteilen gebrochen werden können.

2.1.1. Cäsar. Das Verfahren von Cäsar ist eines der ersten Verschlüsselungsverfahren, deren Einsatz historisch belegt ist. Mitteilungen werden verschlüsselt, indem jeder Buchstabe durch das Zeichen eine feste Anzahl von Positionen später im Alphabet ersetzt wird. Cäsar selbst verwendete eine Verschiebung um drei Zeichen, wechselte also seinen Schlüssel nicht:

$$\begin{pmatrix} \texttt{A} & \texttt{B} & \texttt{C} & \dots & \texttt{X} & \texttt{Y} & \texttt{Z} \\ \texttt{D} & \texttt{E} & \texttt{F} & \dots & \texttt{A} & \texttt{B} & \texttt{C} \end{pmatrix}$$

Das Cäsar-Verfahren kennt 26 verschiedene Schlüssel, dabei ist der Schlüssel 0 trivial, da er auf die identische Abbildung führt. Entschlüsselung und Verschlüsselung verwenden das gleiche Verfahren, jedoch mit verschiedenen Schlüsseln: Ist $0 \leq e < 26$ der Verschlüsselungsschlüssel, also die Verschiebung um e Zeichen, so kann mit $d = 26 - e$ entschlüsselt werden, indem die Zeichen um d im Alphabet geschoben werden. Die Verschlüsselung mit dem Schlüssel $e = 13$ nimmt eine besondere Stellung ein: Die Entschlüsselung erfolgt mit der gleichen Funktion wie die Verschlüsselung: $E_{13} = \mathrm{id}$. Im Internet ist E_{13} auch als `rot13` bekannt, es ist sehr beliebt um zum Beispiel die Antwort auf ein Rätsel so zu verschleiern, damit man sie nicht sofort lesen kann.

Das Cäsar-Verfahren kann sehr leicht mit Hilfe von Buchstabenstatistik gebrochen werden. Da im Deutschen (und auch im Lateinischen) der Buchstabe `E` deutlich häufiger vorkommt als jeder andere Buchstabe, genügt es, das häufigste Zeichen x des Chiffrates zu ermitteln. Der Schlüssel ist diejenige Zahl, die `E` auf x abbildet: $E_e(\texttt{E}) = x$.

Der nachstehende Text wurde mit dem Cäsar-Verfahren verschlüsselt:

```
Niwyw oeq mr wimri Limqexwxehx yrh pilvxi hmi Qirwglir hsvx mr
hiv Wcrekski.  He wxeyrxir eppi yrh wekxir: Asliv lex iv hmiwi
Aimwlimx yrh Ovejx, Ayrhiv dy xyr? Mwx hew rmglx hiv Wslr hiw
Dmqqivqerrw? Limwwx rmglx wimri Qyxxiv Qevme, yrh wmrh rmglx
Neosfyw, Nswij, Wmqsr yrh Nyhew wimri Fvyihiv?  Pifir rmglx wimri
Wglaiwxivr yrxiv yrw? Asliv epws lex iv hew eppiw?  Yrw wmi
relqir Erwxsww er mlq yrh pilrxir mlr ef. He wekxi Niwyw dy
mlrir: Rmvkirhw lex imr Tvstlix ws airmk Erwilir ami mr wimriv
Limqex yrh mr wimriv Jeqmpmi. Yrh aikir mlviw Yrkpeyfirw xex
iv hsvx ryv airmki Ayrhiv.

Eyw hiq Izerkipmyq regl Qexleiyw, 13,54-58
```

Die Buchstabenstatistik

```
A:  2.02   B:  0.00   C:  0.20   D:  0.61   E:  7.49   F:  1.01   G:  1.42
H:  5.47   I: 15.18   J:  0.61   K:  2.02   L:  5.47   M:  7.89   N:  1.01
O:  0.61   P:  2.23   Q:  3.24   R: 12.75   S:  2.63   T:  0.40   U:  0.00
V:  5.26   W:  9.72   X:  6.68   Y:  5.87   Z:  0.20
```

macht sehr schnell sichtbar, dass `I` mit über 15% der häufigste Buchstabe ist, also ist die Wahrscheinlichkeit, dass der Text E_4 verschlüsselt worden ist, sehr gross. Die Entschlüsselung mit E_{22} macht den Abschnitt aus dem Mathäus-Evangelium wieder verständlich:

```
Jesus kam in seine Heimatstadt und lehrte die Menschen dort in
der Synagoge.  Da staunten alle und sagten: Woher hat er diese
Weisheit und Kraft, Wunder zu tun? Ist das nicht der Sohn des
Zimmermanns? Heisst nicht seine Mutter Maria, und sind nicht
Jakobus, Josef, Simon und Judas seine Brueder?  Leben nicht seine
Schwestern unter uns? Woher also hat er das alles?  Uns sie
nahmen Anstoss an ihm und lehnten ihn ab. Da sagte Jesus zu
ihnen: Nirgends hat ein Prophet so wenig Ansehen wie in seiner
Heimat und in seiner Familie. Und wegen ihres Unglaubens tat
```

```
er dort nur wenige Wunder.

Aus dem Evangelium nach Mathaeus, 13,54-58
```

Für die Buchstabenstatistik sind wir von der Kenntnis der Sprache des Textes ausgegangen. Im nachfolgenden Beispiel ist die Sprache des Fragmentes nicht bekannt, doch ist bekannt, dass von den Helvetiern, und deren Fürst, von Orgetorix, die Rede ist:

```
Nchq Uryirgvbf ybatr abovyvffvzhf shvg rg qvgvffvzhf
Betrgbevk. Vf Z.  Zrffnyn, Z. Cvfbar pbafhyvohf ertav phcvqvgngr
vaqhpghf pbavhengvbarz abovyvngvf srpvg rg pvivgngv crefhnfvg
hg qr svavohf fhvf phz bzavohf pbcvvf rkverag: cresnpvyr
rffr, phz iveghgr bzavohf cenrfgnerag, gbgvhf Tnyyvnr vzcrevb
cbgvev. Vq ubp snpvyvhf vvf crefhnfvg, dhbq haqvdhr ybpv anghen
Uryirgvv pbagvaraghe: han rk cnegr syhzvar Eurab yngvffvzb
ngdhr nygvffvzb, dhv ntehz Uryirgvhz n Treznavf qvivqvg;
nygren rk cnegr zbagr Vhen nygvffvzb, dhv rfg vagre Frdhnabf rg
Uryirgvbf; gregvn ynph Yrznaab rg syhzvar Eubqnab, dhv cebivapvnz
abfgenz no Uryirgvvf qvivqvg. Uvf erohf svrong hg rg zvahf yngr
intneraghe rg zvahf snpvyr svavgvzvf oryyhz vasreer cbffrag; dhn
rk cnegr ubzvarf oryynaqv phcvqv zntab qbyber nqsvpvronaghe. Ceb
zhygvghqvar nhgrz ubzvahz rg ceb tybevn oryyv ngdhr sbegvghqvavf
nathfgbf fr svarf unorer neovgenonaghe, dhv va ybatvghqvarz
zvyvn cnffhhz PPKY, va yngvghqvarz PYKKK cngronag.


Nhf qrz refgra Ohpu qrf tnyyvfpura Xevrtrf, Qr Oryyb Tnyyvpb,
iba Whyvhf Pnrfne
```

Da die Verschlüsselung die Wortgrenzen nicht verschleiert, und in obiger Darstellung sogar noch die Eigennamen an der Grossschreibung[1] erkennbar bleiben, können die Kandidaten für das Wort `Orgetorix` sehr leicht ermittelt werden. Es wird ein neun Zeichen langes, gross geschrie-

[1] Geht man von der Voraussetzung aus, dass der Text Deutsch ist, müssten auch andere Nomina gross geschrieben sein. Da aber zum Beispiel der zweitletzte Satz kein einziges gross geschriebenes Wort hat, muss man wohl annehmen, dass der Text nicht Deutsch ist. Darauf deuten auch die Anhäufungen von Grossbuchstaben in der letzten Zeile hin.

benes Wort gesucht:

```
Uryirgvbf
Betrgbevk
Uryirgvhz
Uryirgvvf
```

In dieser Liste wurden Duplikate bereits entfernt. Die Lösung muss an der ersten und sechsten sowie an der zweiten und siebten Stelle das gleiche Zeichen enthalten, dies erlaubt nur noch die Lösung `Betrgbevk`. Alternativ kann man auch argumentieren, dass die nichttriviale Cäsar-Verschlüsselung niemals einen Buchstaben auf sich selbst abbildet (sie müsste dann auch alle anderen Zeichen auf sich selbst abbilden, wäre also die triviale Verschlüsselung E_0), die Kandidaten, die an der zweiten Stelle ein `r` haben, können also nicht Verschlüsselungen des Wortes `Orgetorix` sein[2]. Somit wird der Buchstabe `O` auf `B` abgebildet, dies ist genau E_{13}. Die Entschlüsselung, ebenfalls mit E_{13} enthüllt, dass der Klartext

```
Apud Helvetios longe nobilissimus fuit et ditissimus
Orgetorix. Is M.  Messala, M. Pisone consulibus regni cupiditate
inductus coniurationem nobiliatis fecit et civitati persuasit
ut de finibus suis cum omnibus copiis exirent: perfacile
esse, cum virtute omnibus praestarent, totius Galliae imperio
potiri. Id hoc facilius iis persuasit, quod undique loci natura
Helvetii continentur: una ex parte flumine Rheno latissimo
atque altissimo, qui agrum Helvetium a Germanis dividit;
altera ex parte monte Iura altissimo, qui est inter Sequanos et
Helvetios; tertia lacu Lemanno et flumine Rhodano, qui provinciam
nostram ab Helvetiis dividit. His rebus fiebat ut et minus late
vagarentur et minus facile finitimis bellum inferre possent; qua
ex parte homines bellandi cupidi magno dolore adficiebantur. Pro
multitudine autem hominum et pro gloria belli atque fortitudinis
angustos se fines habere arbitrabantur, qui in longitudinem
milia passuum CCXL, in latitudinem CLXXX patebant.


Aus dem ersten Buch des gallischen Krieges, De Bello Gallico,
von Julius Caesar
```

[2] Man nennt dieses Schlussverfahren, bei dem man eine Möglichkeit wegen nicht erlaubter Übereinstimmungen ausschliesst, negative Mustererkennung.

lateinisch war.

Dieses Beispiel illustriert zwei Schwächen der oben dargestellten Verschlüsselung, sowie die kryptanalytischen Verfahren, die daraus resultieren.

1. Die Verteilungen der Buchstaben in Klartext und Chiffrat sind identisch, nur die Zeichen sind verschoben. Ein gutes Verfahren sollte auch die Häufigkeitsverteilung der Buchstaben so verändern, dass jedes Zeichen gleich häufig ist.
2. Das Verfahren erfasst nur einzelne Zeichen, und beeinflusst die "grossräumigere" Struktur wie Wortgrenzen innerhalb des Klartextes nicht. Ein verbessertes Verfahren sollte Wortgrenzen nicht mehr zeigen, und möglichst mehreren Zeichen einen Einfluss auf jedes Output-Zeichen geben.

2.1.2. Vigenère. Das Vigenère-Verfahren verbessert das Cäsar-Verfahren im Bezug auf die erste Schwäche: statt eines einzelne Cäsar-Schlüssels wird der Schlüssel bei jedem Zeichen gewechselt. Zeichen i wird also mit E_{e_i} verschlüsselt. Die Folge der Zahlen e_i bestimmt das Verfahren eindeutig, sie ist der Schlüssel. Aus praktischen Gründen wird eine periodische Folge verwendet, sie wiederholt sich nach N Teilschlüsseln: $e_{i+N} = e_i$ für alle i. Wir schreiben $E_{e_1,\ldots,e_N}$ für die Verschlüsselung mit der Folge $e_1, \ldots, e_N$.

Durch geeignete Wahl der Folge e_i kann der Angriff über die Schwäche 1 auf den ersten Blick verunmöglicht werden. Ist $N = 26$ und ist die Folge $e_1, \ldots, e_{26}$ eine Permutation der Zahlen $1, \ldots, 26$, dann ist die Wahrscheinlichkeitsverteilung eines sehr grossen, mit $E_{e_1,\ldots,e_{26}}$ verschlüsselten Textes identisch mit Verschlüsselung durch $E_{1,\ldots,26}$. Diese wiederum ist das Mittel der Verteilungen von $E_1, \ldots, E_{26}$, die jedoch alle bis auf eine Verschiebung identisch sind. Somit ist das Mittel eine Gleichverteilung.

In der Praxis werden jedoch meistens geringere Perioden verwendet, so dass ein Angriff über Buchstabenstatistik nicht ausgeschlossen ist. Im Folgenden Bespiel wurde ein deutscher Text nach Vigenère verschlüsselt:

```
Fu xcu gloodm hjp bnwfu Tekmqvt pjvwfp jp fkqfo htrtuho
gjenfp Xcoe, ecujpqfp xqkovho hjph coug Gtdv jbpc comglo,
ecv yds hjph Guabdvdhskq.  Dn Wbih oddjwf vjg tkfi cvt Lcwag
vpg bxs Qbekugxmg, fht Dcgqeu bdhs zvtgf vjg xkhegu quegqunldj
```

```
xkh glo Pfpvdj hgvucougw.  Tkh mropwf gbu Xkoe xof ekh Xrfihm
kftefkopenfp, wqe gbpq ufinddjwfvh ulf, lqfivh wqe eskhu ht.
Yhop kgpbpg cxg kvpgftw Ufitluvh fhn Vdjopuv pdignbo, ur oxfuvug
ft tvlmnhtvhigq wqe nppqug tkfi qjeku ypp egu Uwfnof efyhhgq,
eju tkh kko opuvqtddj; yhop bdhs hjph mhvufig Kwqhhubw jp ekhtgq
Mufkv mdn, tq wguxcqegoug tkh flfuhmdh kq glogq Xrhgo wqe vqgusvh
ulf gbpq kq glogp Mrsd fkq wqe wswj fho Npte kq glog Lcpngu fht
Vdjopuvfu. Ulf kbvwf zpjo ulfdhovdvuhof tqodjhs Npgucg nkw ur
tdsgq Xrfihmp jo Tekmqvtg.

Mptlofh wqe Mptloihm, bwv Iujop'u Nchsekfp
```

Die Buchstabenstatistik

A:	0.30	B:	2.42	C:	2.42	D:	4.08	E:	4.38	F:	7.10	G:	7.55
H:	7.85	I:	1.96	J:	3.93	K:	4.98	L:	2.87	M:	2.57	N:	2.57
O:	6.04	P:	6.50	Q:	5.44	R:	1.21	S:	1.81	T:	4.68	U:	7.25
V:	5.14	W:	3.63	X:	2.27	Y:	0.76	Z:	0.30				

zeigt keinen eindeutigen Kandidaten mehr, `F`, `G`, `H` und `U` kommen alle etwa gleich häufig vor. Man könnte versucht sein, daraus auf die Periode vier zu schliessen, doch lässt sich mit diesen Mitteln noch nicht auf die einzelnen Teilschlüssel e_i schliessen.

Trotz der besseren Zeichenstatistik ist Vigenère immer noch sehr leicht angreifbar. Sobald die Periode N bekannt ist, kann das Chiffrat C in die Teiltexte

$$C_i = (c_i, c_{i+N}, c_{i+2N}, \dots)$$

zerlegt werden, die mit dem Cäsar-Schlüssel E_{e_i} verschlüsselte Teiltexte sind. Auf sie ist also die normale Statistik anwendbar, mit deren Hilfe E_{e_i} ermittelt werden kann.

Die Periode N kann zum Beispiel dadurch ermittelt werden, dass mit verschiedenen Werten für N die Buchstabenstatistik im Text

$$(c_1, c_{1+N}, c_{1+2n}, \dots)$$

aufgestellt wird, und N als derjenige Wert gewählt wird, bei dem die Statistik am ehesten der zu erwarteten Verteilung entspricht.

Autokorrelation. Der Vergleich zweier Verteilungen ist keine triviale

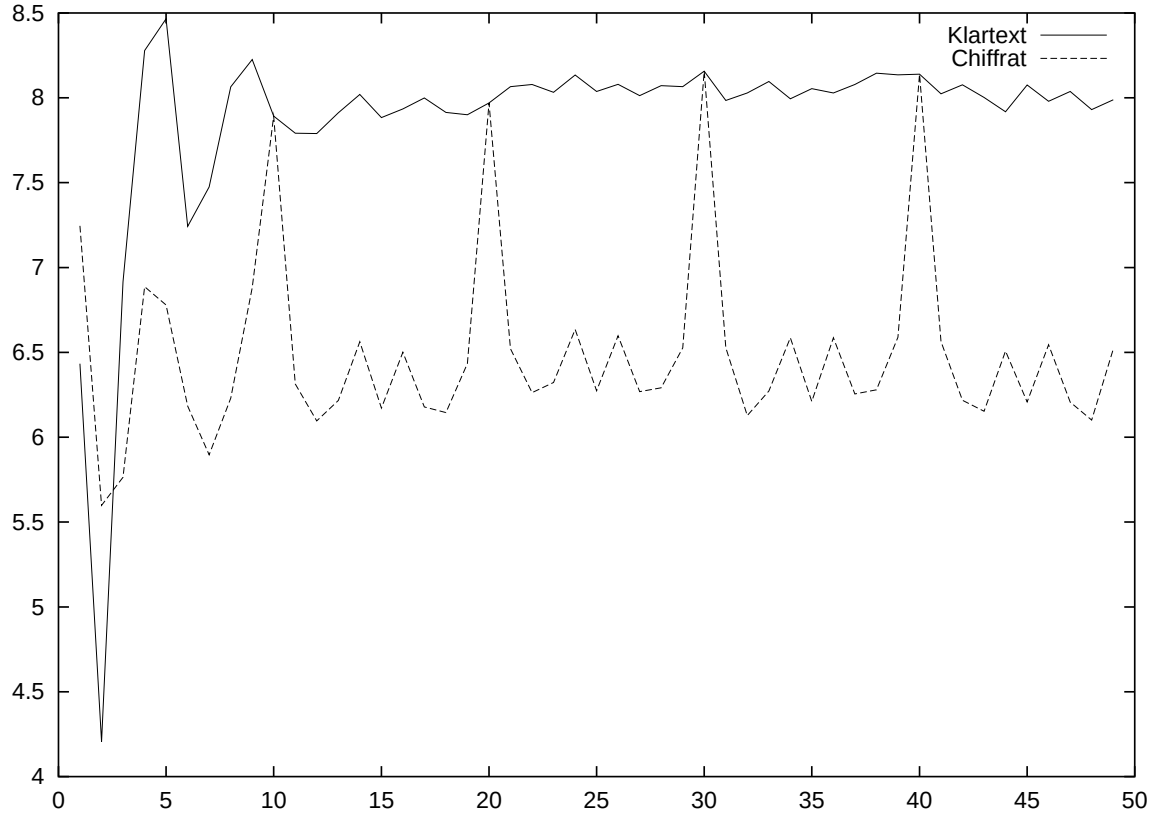

Abbildung 2.1. Autokorrelation in Klartext und Chiffrat.

Aufgabe, und führt auf aufwendige Rechnungen. Mit Hilfe von Autokorrelation eines Textes kann die Periode N sehr viel leichter ermittelt werden.

Ein natürlichsprachlicher Text hat eine gewisse Periodizität, Silben haben ähnliche Längen, die Abfolge von Zeichen wiederholt sich oft. Vergleicht man einen Text t der Länge n mit einer verschobenen Kopie seiner selbst, und zählt die übereinstimmenden Zeichen, ergibt sich ein Wert, der im wesentlichen unabhängig ist von der Verschiebung. Wir definieren

$$\kappa(d,t) = \frac{1}{n-d}|\{0 < i < n-d | t_i = t_{i+d}\}|.$$

Für einen englischen Text von ca 300kB findet man beispielsweise die Werte in Abbildung 2.1. (ausgezogene Kurve). Man sieht auch, dass identische Zeichen ausgesprochen selten nebeneinander stehen: es ist viel wahrscheinlicher, dass ein Buchstabe drei Stellen später nochmals vor-

kommt als eine oder zwei Stellen später.

Bei einem Cäsar-verschlüsselten Text ändert sich der Wert von κ nicht, da E_e auf alle Zeichen unabhängig von deren Position im Klartext gleich wirkt. Da κ nur eine statistische Aussage über Buchstabenfolgen ist, kann man die gleichen Aussagen für Teiltexte verlangen. So kann $\kappa(d, \cdot)$ mit Hilfe der Teiltexte $t_{d,i} = (c_i, c_{i+d}, c_{i+2d}, \dots)$ berechnet werden:

$$\kappa(d, t) = \frac{1}{|t| - d} \sum_{i=1}^{d} \kappa(d, t_{d,i})(|t_{d,i}| - 1)$$

Solange der Offset d die Periode N nicht teilt, werden bei der Bestimmung von $\kappa(d, t)$ Texte miteinander verglichen, die mit ein keiner Weise zusammenhängenden Cäsarschlüsseln verschlüsselt wurden. Man kann nicht erwarten, dass der Wert in irgend einer Beziehung zum erwarteten Wert des Klartextes steht. Nur falls der Offset d gerade ein Vielfaches der Periode ist, erwartet man die gleichen Werte wie beim Klartext, da in diesem Falle immer gleichartig verschlüsselte Teiltexte verglichen werden. Tatsächlich ist der Wert von $\kappa(d, t)$ klar kleiner, wenn d kein Vielfaches von N ist, wie Abbildung 2.1. mit nicht zu überbietender Deutlichkeit zeigt.

Damit ist auch ein allgemeines Verfahren zur Entschlüsselung von Vigenère gefunden: zunächst berechnet man die Werte von $\kappa(d, t)$ und bestimmt aus den sich abzeichnenden Maxima die Periode N, anschliessend verwendet man Buchstabenstatistik in den Teiltexten $t_i, 1 \leq i < N$ zur Ermittelung der Teilschlüssel e_i.

Beispiel. Für den oben bereits zitierten Text deuten die Werte von κ an, dass die Periode $N = 3$ ist:

d	κ	d	κ	d	κ
1	1.594533	9	6.666667	17	5.800464
2	2.166477	10	3.682394	18	7.549361
3	5.022831	11	5.760369	19	4.651163
4	6.628571	12	6.228374	20	3.492433
5	4.919908	13	5.196305	21	7.226107
6	6.300115	14	4.508671	22	4.900817
7	3.899083	15	8.333333	23	2.920561
8	5.281286	16	3.707995	24	6.900585

Mit der Buchstabenstatistik in den Teiltexten findet man, dass die Folge der Teilschlüssel $(1, 2, 3)$ ist. Tatsächlich ergibt die Entschlüsselung mit $(25, 24, 23)$ den offensichtlich korrekten Klartext

```
Es war einmal ein altes Schloss mitten in einem grossen
dicken Wald, darinnen wohnten eine alte Frau ganz allein,
das war eine Erzzauberin.  Am Tage machte sie sich zur Katze
und zur Nachteule, des Abends aber wurde sie wieder ordentlich
wie ein Mensch gestaltet.  Sie konnte das Wild und die Voegel
herbeilocken, und dann schlachtete sie, kochte und briet es.
Wenn jemand auf hundert Schritte dem Schloss nahekam, so muesste
er stillestehen und konnte sich nicht von der Stelle bewegen,
bis sie ihn lossprach; wenn aber eine keusche Jungfrau in diesen
Kreis kam, so verwandelte sie dieselbe in einen Vogel und sperrte
sie dann in einem Korb ein und trug den Korb in eine Kammer des
Schlosses. Sie hatte wohl siebentausend solcher Koerbe mit so
raren Voegeln im Schlosse.

Jorinde und Joringel, aus Grimm's Maerchen
```

2.1.3. Allgemeine Substitution. Man könnte eine Verbesserung des Cäsar-Verfahrens auch dadurch anstreben, dass die Verschlüsselung eines einzelnen Buchstabens nicht bereits den Schlüssel bestimmt. Statt der zyklischen Vertauschung könnte eine beliebige Permutation $\sigma \in S_{26}$ der 26 Buchstaben des Alphabetes gewählt werden. Die Auswahl an Permutationen ist wesentlich grösser: $26! \simeq 4.03 \cdot 10^{26}$, also von der Grössenordnung der Avogadroschen Zahl!

Allerdings ist auch dieses Verfahren mit statistischen Methoden angreifbar. Da alle Zeichen mit der gleichen Substitution verschlüsselt werden, lassen sich die häufigsten Zeichen schnell identifizieren. Meist werden für einen menschlichen Leser bereits einige Wörter erkennbar, wenn die häufigsten drei Buchstaben entschlüsselt sind, und erlauben ihm, weitere Zeichen zu erraten.

So ist zum Beispiel im Text

```
Kcn asxcgwsg mgu ct Rqdns 50 ibn Zdncujmu. Eqgp Eqvvcsg cuj ibg
wsg Nbstsng asusjpj. Eqgp Eqvvcsg? Gscg! Scg ibg mgasmeuqtsg
Eqvvcsng asibsvfsnjsu Wbnx dbsnj gczdj qmx, wst Scgwncgevcge
Kcwsnujqgw pm vscujsg. Mgw wqu Vsasg cuj gczdj vsczdj xmsn wcs
nbstcuzdsg Vsecbgqsns, wcs qvu Asuqjpmge cg wsg asxsujcejsg
```

```
Vqesng Aqaqbnmt, Qhmqncmt, Vqmwqgmt mgw Fvscgabgmt vcsesg.
```

sehr rasch erkennbar, dass der häufigste Buchstabe das `S` ist. Im Deutschen gibt es ausserdem nur wenige Wörter mit drei Buchstaben, die mit `E` beginnen, bei `scg` auf der zweiten Zeile kann es sich also fast nur um `ein` handeln. Mit diesem Wissen wird der Text zu

```
_i_ _e_in_en _n_ i_ ____e 50 ___ ___i____. __n_ ____ien i__ __n
_en __e_e_n _e_e___. __n_ ____ien? Nein! Ein __n _n_e_____en
____ie_n _e__e__e__e_ ____ __e__ ni___ ___, _e_ Ein__in__in_
_i_e____n_ __ _ei__en. _n_ ___ _e_en i__ ni___ _ei___ __e_ _ie
__e_i___en _e_i_n_e_e, _ie ___ _e_____n_ in _en _e_e__i__en
___e_n ________, _____i__, _____n__ _n_ __ein__n__ _ie_en.
```

Mit ähnlichen Überlegungen lässt sich auch in der dritten Zeile die Zeichenfolge `gczdj` als `nicht` identifizieren:

```
_i_ _e_in_en _n_ i_ __h_e 50 ___ Ch_i_t__. __n_ ____ien i_t __n
_en __e_e_n _e_et_t. __n_ ____ien? Nein! Ein __n _n_e_____en
____ie_n _e__e__e_te_ ____ h_e_t nicht ___, _e_ Ein__in__in_
_i_e__t_n_ __ _ei_ten. _n_ ___ _e_en i_t nicht _eicht __e_ _ie
__e_i_chen _e_i_n_e_e, _ie ___ _e__t__n_ in _en _e_e_ti_ten
___e_n ________, _____i__, _____n__ _n_ __ein__n__ _ie_en.
```

In diesem Stile fortfahrend, wird der aufmerksame Kryptanalytiker bald die Anfangssätze der Asterix-Abenteuer wiedererkennen:

```
Wir befinden uns im Jahre 50 vor Christus. Ganz Gallien ist von
den Roemern besetzt. Ganz Gallien? Nein! Ein von unbeugsamen
Galliern bevoelkertes Dorf hoert nicht auf, dem Eindringling
Widerstand zu leisten. Und das Leben ist nicht leicht fuer die
roemischen Legionaere, die als Besatzung in den befestigten
Lagern Babaorum, Aquarium, Laudanum und Kleinbonum liegen.
```

Offensichtlich ist auch eine allgemeine Substitution mit statistischen Hilfsmitteln und unter Kenntnis der Sprache das Klartextes durchaus zu brechen. Selbst bei einer Verkomplizierung durch wechselnde Subsitutionen von Zeichen zu Zeichen ähnliche wie bei Vigenère liessen sich die dort besprochenen Verfahren weiterhin anwenden. Für gute Kryptographie ist also wesentlich mehr notwendig. Die nachfolgenden Abschnitte sollen einerseits zeigen, wo der Hebel zur Verbesserung anzusetzen ist, und später, wie dies genau geschehen kann.

2.2. Theoretische Grundlagen.

2.2.1. Shannon's Informationstheorie. In der Zeit nach dem zweiten Weltkrieg wurden die theoretischen Grundlagen der Kryptographie durch Shannon's Informationstheorie gelegt. Leider sind die darauf aufbauenden Resultate fast alle klassifiziert. Wichtig für unsere Betrachtungen ist jedoch die Beobachtung, dass ein Verschlüsselungsverfahren offensichtlich als sicher gelten kann, wenn das Chiffrat keinerlei Information über den Klartext enthält, eine Aussage, die mit Hilfe informationstheoretischer Mittel untersucht werden kann.

2.2.2. One-Time Pad. Der als aussichtslos schlecht erscheinende Vigenère Algorithmus verschleiert die Information über die Periode im Chiffrat nicht ausreichend, die Funktion κ ist in der Lage, die Periode sichtbar zu machen. Wenn es aber gar keine Periode gibt, die Verschlüsselung mit Hilfe einer unendlichen Folge von Teilschlüsseln $(e_i)_{i\in\mathbb{N}}$ geschieht, die völlig zufällig sind, und auch den Cäsar-Schlüssel 0 nicht auslassen, lässt sich aus dem so entstehenden Chiffrat

$$(E_{e_i}(t_i))_{i\in\mathbb{N}}$$

nichts mehr über den Klartext herauslesen. Andernfalls könnte man daraus nämlich eine Gesetzmässigkeit der Teilschlüssel e_i ableiten, was deren Zufälligkeit widersprechen würde.

In der binären Welt der Praxis wird dieses Verfahren nicht mit Cäsars Verfahren als Basis aufgebaut. Stattdessen wird zu jedem Byte des Klartext, einer Folge von Bytes $(t_1, \dots)$, je ein Bytes einer zufälligen Folge von Schlüsselbytes (k_i) mit XOR hinzuaddiert: $c_i = t_i \oplus k_i$.

Dieses so genannte One-Time Pad ist im Sinne von Shannons Informationstheorie absolut sicher. Es ist jedoch nur beschränkt praktikabel, da eine grosse Menge von Schlüsselmaterial (die Bytes (k_i)) notwendig ist, dass beiden Parteien bekannt sein muss. In der Praxis wird das One-Time Pad daher meist mit Hilfe einer Pseudozufallsfolge realisiert, welche von einem nur den beiden Partnern bekannten Startwert abhängt.

2.2.3. Grundprinzipien. Offensichtlich genügt es nicht, nur die Bedeutung eines einzelnen Zeichens zur verändern. Der Schritt vom Cäsar- zum Vigenère-Verfahren hat gezeigt, wie mit positionsahängiger Verschlüsselung eine Verbesserung der statistischen Eigenschaften des Ge-

sammttextes erreicht werden konnte. Wir werden diese Idee später nochmals treffen, wenn wir die Verschlüsselung einzelner Datenblöcke (den elektronischen Codebook Mode ECB) von der Position innerhalb des Datenstromes abhängig machen mit Hilfe von Feedback.

Das Vigenère-Verfahren hat die Verschlüsselung von seiner Position im Text abhängig gemacht, aber offensichtlich noch nicht stark genug. Wirklich starke Verfahren müssen in wesentlich weiter gehendem Masse die Verschlüsselung eines Einzelzeichens von dessen Position und Umgebung abhängig machen.

Shannon hat darauf hingewiesen, dass für eine gute Verschlüsselung eine möglichst ausgewogene Mischung der folgenden zwei Prinzipien erforderlich ist:

Konfusion. Unter Konfusion versteht man möglichst grosse Zufälligkeit zwischen den Eingabezeichen und dem zugehörigen Output. Cäsar und Vigenère unterscheiden sich bezüglich Konfusion überhaupt nicht. Eine allgemeine Substitution verbessert die Konfusion gegenüber einem einfachen Cäsar-Verfahren.

Diffusion. Die Verschlüsselung eines Zeichens soll in gleichmässiger Weise nicht nur vom Klartextzeichen selbst, sondern auch von allen seinen Nachbarn bestimmt sein. Alle bisher besprochenen Verfahren besitzen überhaupt keine Diffusion.

Kapitel 3. Mathematische Hilfsmittel

In diesem Kapitel werden die mathematischen Hilfsmittel zum Verständnis der modernen kryptographischen Verfahren zusammengestellt. Dabei geht es in erster Linie um die verschiedenen möglichen Multiplikationen, die man auf Zahlen oder Polynomen definieren kann. Besonders interessant sind dabei die Polynome, denn Polynome mit Koeffizienten in $\mathbb{F}_2 = \{0, 1\}$ können sehr maschinenfreundlich als Bitvektoren dargestellt werden. Deren Addition ist das bitweise XOR, was von aktuellen Prozessoren sehr schnell und gut parallelisierbar (da ohne Übertrag) ausgeführt werden kann.

Allen diesen Operationen liegt die allgemeine Struktur eines Ringes zu Grunde, die im ersten Abschnitt beschrieben wird. Der zweite Abschnitt behandelt die zahlentheoretischen Hilfsmitteln, insbesondere die Definition von Primfaktoren und den Satz über die eindeutige Faktorzerlegung.

3.1. Ringe.

Die Menge der ganzen Zahlen $\mathbb{Z}$ ist die Bühne der elementaren Zahlentheorie: das Fehlen einer multiplikativen Inversen, also das Problem, dass nicht jede Division "aufgeht", macht den Begriff der Teilbarkeit und die Reste zu interessanten Themen. Diese sind jedoch nicht nur bei $\mathbb{Z}$ anzutreffen, im Gegenteil. Wir finden hier eine allgemeine Struktur, die in verschiedenen algebraischen Zusammenhängen immer wieder auftaucht, und auch in der Kryptographie von Nutzen ist, nämlich die Struktur eines Ringes.

3.1.1. Definition. Charakteristisch an $\mathbb{Z}$ sind die Addition, die immer umkehrbar ist, und die Multiplikation, die nicht immer eine Inverse besitzt. Die Definition des Ringes formuliert genau dies in abstrakter Form:

Definition 3.1 (Ring). *Eine Menge R mit zwei Operationen Addition und Multiplikation heisst ein Ring, wenn die folgenden Eigenschaften erfüllt sind:*

- R1) *Die Addition ist assoziativ:* $(a+b)+c = a+(b+c) \quad \forall a, b, c \in R$.
- R2) *Die Addition ist kommutativ:* $a + b = b + a \quad \forall a, b \in R$.
- R3) *Es gibt ein neutrales Element* 0 *der Addition:* $a + 0 = 0 + a = 0 \quad \forall a \in R$.

R4) *Zu jedem Element $a \in R$ gibt es ein additives Inverses $(-a)$:* $a + (-a) = 0 \quad \forall a \in R$.

R5) *Die Multiplikation ist assoziativ:* $(ab)c = a(bc) \quad \forall a, b, c \in R$.

R6) *Es gibt ein neutrales Element der Multiplikation* 1*:* $1 \cdot a = a \cdot 1 = a \quad \forall a \in R$.

R7) *Addition und Multiplikation vertragen sich gemäss dem Distributivgesetz:*

$$a(b + c) = ab + ac$$
$$(a + b)c = ac + bc$$

für alle $a, b, c \in R$.

Die ersten vier Forderungen drücken aus, dass R mit der Addition eine abelsche Gruppe ist. Die Ringe in den Anwendungen werden zudem eine kommutative Multiplikation haben, man spricht in diesem Fall von einem kommutativen Ring.

3.1.2. Beispiele. Ein Grossteil der klassischen Algebra spielt sich in Ringen ab:

1. Die Menge der natürlichen Zahlen $\mathbb{N}$ ist kein Ring, weil die Forderung R4, die Existenz eines additiven Inversen, nicht erfüllt ist. $\mathbb{N}$ ist nicht einmal eine Gruppe.
2. $\mathbb{Z}$ ist ein Ring.
3. $\mathbb{Q}$, $\mathbb{R}$ und $\mathbb{C}$ sind Ringe, in denen zusätzlich die zu jedem von 0 verschiedenen Element eine multiplikative Inverse existiert. Ein derartiger Ring heisst ein *Körper*.
4. Die $n \times n$-Matrizen mit Elementen in einem Körper K werden $M_n(K)$ geschrieben und bilden einen (nicht kommutativen) Ring. Es ist nicht möglich, sich auf die invertierbaren Matrizen zu beschränken, um so vielleicht eine Körperstruktur zu erreichen, denn die Summe zweier invertierbarer Matrizen braucht nicht invertierbar zu sein, wie das folgende Beispiel zeigt:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}.$$

Im Falle $n = 1$ fällt der Ring mit K zusammen, ist dann also kommutativ und sogar ein Körper.

Weitere Beispiele lassen sich aus diesen Ringen Konstruieren.

3.1.3. Polynomringe. Ein weiteres Beispiel lässt sich mit Hilfe von Polynomen konstruieren. Zu einem Ring R betrachten wir die Polynome mit Koeffizienten in R, also formale Ausdrücke der Form

$$a_n x^n + \dots a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

und bezeichnen die Menge dieser Polynome mit

$$R[x] = \{a_n x^n + \dots a_{n-1} x^{n-1} + \dots + a_1 x + a_0 | a_i \in R, 0 \leq i \leq n\}.$$

Die übliche Addition und Multiplikation macht daraus einen Ring:

$$(a_n x^n + \dots a_0) + (b_n x^n + \dots b_0) = (a_n + b_n) x^n + \dots + (a_0 + b_0)$$

$$(a_n x^n + \dots a_0) + (b_n x_n + \dots b_0) = \sum_{i=0}^{n+m} \left(\sum_{j=0}^{\min(i,n)} a_j b_{i-j} \right) x^i$$

wobei in diesen Formeln auch erlaubt ist, dass die beiden Polynome $a = a_n x^n + \dots a_0$ und $b = b_n x^n + \dots b_0$ verschiedenen Grad haben, also dass einige der Koffizienten mit grossem Index von a oder b verschwinden.

3.2. Ideale und Restklassen.

Die ganzen Zahlen $\mathbb{Z}$ kann man in Klassen einteilen, die alle den gleichen Rest bei Teilung durch eine feste Zahl n haben. Solche Zahlen unterscheiden sich um ein Vielfaches von n, die Klassen sind also Teilmengen der Form

$$r + n\mathbb{Z} = \{r + kn | k \in \mathbb{Z}\},$$

wobei $0 \leq r < n$ der allen Zahlen der Klasse gemeinsame Rest ist, den man bei Division durch n erhält. Offensichtlich ist durch die Zahl r die Klasse eindeutig bestimmt, wir schreiben daher für die Menge der Klassen einfach

$$\mathbb{Z}/n\mathbb{Z} = \{0, 1, \dots, n-1\}.$$

Die arithemtischen Operationen auf $\mathbb{Z}$ lassen sich direkt auf $\mathbb{Z}/n\mathbb{Z}$ übertragen. Die Summe oder das Produkt von zweier Klassen wird dabei berechnet, indem alle Elemente der beiden Klasse miteinander verknüpft werden. Zum Beispiel für die Addition:

$$r + kn + s + ln = (r + s) + (k + l)n.$$

Falls $r + s$ nicht zwischen 0 und n liegt, muss n subtrahiert werden. die Wahl von k und l spielt offensichtlich keine Rolle. Analog gilt für die Multiplikation

$$(r + kn)(s + ln) = rs + (rl + ks + kln)n,$$

unabhängig von der Wahl von k und l wird das Produkt immer in der gleichen Klasse sein. Falls rs nicht zwischen 0 und n liegt, muss zunächst der Rest bei Division durch n ermittelt werden.

Diese Konstruktion der Restklassen in $\mathbb{Z}$ funktioniert in jedem Ring. Die Division mit Rest ist dabei nicht unbedingt wohldefiniert (es gibt ja in R im allgemeinen weder eine Division noch einen sinnvollen Teilbarkeitsbegriff), daher muss die Definition leicht modifiziert werden.

Definition 3.2 (Ideal). *Ein Ideal I ist eine Teilmenge von R mit folgenden Eigenschaften:*

- I1) *I ist additiv abgeschlossen: $a + b \in I \quad \forall a, b \in I$.*
- I2) $0 \in I$
- I3) *Die additiven Inversen aller Elemente von I sind ebenfalls in I: $(-a) \in I \quad \forall a \in I$.*
- I4) *Die Multiplikation mit einem Element von R führt nicht aus I heraus: $aI \subset I \quad \forall a \in R$.*

Die ersten drei Forderungen besagen, dass I eine Untergruppe von R (betrachtet als abelsche Gruppe) ist. In I4 ist wichtig, dass die Aussage für alle Elemente a von R gilt, nicht nur für jene von I. Eine multiplikativ abgeschlossene Teilmenge ist also noch kein Ideal, oder nur eine Teilmenge von R, die auch ein Ring ist (ein Teilring) ist nicht notwendigerweise ein Ideal.

Die oben betrachtete Menge $n\mathbb{Z}$ ist ein Ideal in $\mathbb{Z}$. Sie enthält alle Vielfachen von n. Sie ist additiv abgeschlossen (Summen von Elementen in $n\mathbb{Z}$ bleiben in der Menge), denn die Summe zweier Vielfacher n ist ebenfalls ein Vielfaches von n. Multipliziert man ein Vielfaches von n mit irgend einer Zahl, bleibt es natürlich ein Vielfaches von n.

Satz 3.3 (Restklassen bezüglich eines Ideals). *Sei $I \subset R$ ein Ideal in einem kommutativen Ring R. Die Menge der Restklassen*

$$\{a + I | a \in R\}$$

ist eine Ring. Die Ring-Operationen sind

$$\begin{aligned} + &: (a+I, b+I) \mapsto a+b+I \\ \cdot &: (a+I, b+I) \mapsto ab+I \end{aligned}$$

Wir haben bereits gesehen, dass $n\mathbb{Z}$ ein Ideal in $\mathbb{Z}$ ist. Die Menge $a+n\mathbb{Z}$ enthält genau diejenigen Zahlen, die den gleichen Rest bei Teilung durch n haben wie a, das ist also genau die Restklasse von a. Der zu Beginn des Abschnitts beschriebene Ring $\mathbb{Z}/n\mathbb{Z}$ ist also identisch mit dem hier konstruierten Ring.

Beweis: Zunächst ist zu zeigen, dass die Operationen wohldefiniert sind, also nicht von der Wahl der Elemente a und b abhängen. Seien a' und b' andere Repräsentaten, also Elemente von R so, dass $a+I = a'+I$ und $b+I = b'+I$. Deren Differenz muss offensichtlich in I liegen. Folglich gilt

$$\begin{aligned} (a'+I)+(b'+I) &= (a'-a+a+I)+(b'-b+b+I) \\ &= (a+I)+(b+I) \\ (a'+I)(b'+I) &= (a'-a+a+I)(b'-b+b+I) \\ &= \underbrace{(a'-a)(b'-b)}_{\in I} + (a'-a)(b+I) + (b'-b)(a+I) \\ &\quad + (a+I)(b+I) \\ &= \underbrace{(a'-a)}_{\in I} b + (a'-a)I + \underbrace{(b'-b)}_{\in I} a + (b'-b)I \\ &\quad + (a+I)(b+I) \\ &= (a+I)(b+I) \end{aligned}$$

Dabei haben wir für die Produkte $(a'-a)b$ und $(b'-b)a$ die Eigenschaft des Ideals verwendet, $a'-a \in I$ und $b \in I$ haben zur Folge, dass $(a'-a)b \in I$, und analog für $(b'-b)a$.

Die Addition ist selbstverständlich umkehrbar, und die neutralen Elemente von Addition und Multiplikation übertragen sich von R auf die Restklassen. Damit ist der Satz bewiesen.

Definition 3.4 (Restklassenring). *Der im Satz 3.3 konstruierte Ring aus Restklassen bezüglich des Ideals I wird mit R/I bezeichnet.*

3.2.1. Restklassen in $\mathbb{Z}$. Das einfachste Bespiel einer Restklassen-Konstruktion sind die bekannten Reste in $\mathbb{Z}$. Das Ideal $I = n\mathbb{Z}$ besteht aus allen Vielfachen von n. Die Menge $a + n\mathbb{Z}$ besteht aus allen Zahlen, die sich um ein Vielfaches von n von a unterscheiden. Gleichbedeutend damit ist, dass die Zahlen in $a + n\mathbb{Z}$ bei Teilung durch n den gleichen Rest wie a ergeben.

Die Menge $\mathbb{Z}/n\mathbb{Z}$ besteht also aus den Klassen von Zahlen mit gleichem Rest bei Teilung durch n. Da sich der Rest immer als Zahl zwischen 0 und $n-1$ darstellen lässt, kann $\mathbb{Z}/n\mathbb{Z}$ mit der Menge der Reste $\{0, \ldots, n-1\}$ bei Teilung durch n gleichgesetzt werden. Die Operationen sind die Addition und Multiplikation der Reste, gefolgt von der Restbildung modulo n.

Im Falle $n = 5$ erhält man Beispielsweise die folgenden Additions- und Multiplikationstabellen:

+	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

·	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

Man kann daraus ablesen, dass in $\mathbb{Z}/5\mathbb{Z}$ sogar jedes von 0 verschiedene Element bezüglich der Multiplikation invertierbar ist. $\mathbb{Z}/5\mathbb{Z}$ ist also sogar ein Körper, der mit $\mathbb{F}_5$ bezeichnet wird.

Für $n = 6$ ergibt sich jedoch in der Multiplikationstabelle ein anderes Bild:

·	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

Die Produkte eines geraden Restes mit 3 ergeben jeweils 0, entsprechend sind diese Multiplikation nicht unvertierbar: 2, 3 und 4 sind nicht invertierbare Elemente.

3.2.2. Der Körper $\mathbb{F}_2$. Der Ring $\mathbb{Z}/2\mathbb{Z}$ enthält genau zwei Elemente 0 und 1, also das in der Definition eines Ringes geforderte Minimum. Die Operationen sind entsprechend einfach:

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

Die Addition ist also die XOR-Operation, die Multiplikation ist AND. Die nicht sehr zahlreichen Multiplikationen mit nicht verschwindenden Elementen sind alle umkehrbar, $\mathbb{F}_2$ ist also sogar ein Körper.

3.2.3. Restklassen in Polynomringen. Auch in den Polynomringen lässt sich die Restklassenkonstruktion durchführen. Zu einem Polynom $m(x)$ erlaubt er übliche Algorithmus der Polynom-Division Quotienten durch m und vor Allem Reste bei Teilung durch m zu bestimmen. Wie in $\mathbb{Z}$ ist die Menge $m\mathbb{Q}[x] \subset \mathbb{Q}[x]$ bzw. $m\mathbb{F}_p[x] \subset \mathbb{F}_p[x]$ ein Ideal, sie enthält genau die Polynome, die durch $m(x)$ teilbar sind. Allerdings funktioniert der Algorithmus nur dann wirklich, wenn die Koeffizienten in einem Körper gewählt werden, wie wenn man in den Ringen $\mathbb{Q}[x]$ oder $\mathbb{F}_2[x]$ arbeitet.

Als Beispiel betrachten wir den Ring $\mathbb{Q}[x]/(x^2+1)\mathbb{Q}[x]$. Von einem beliebigen Polynom in $\mathbb{Q}[x]$ interessiert nur noch der Rest bei Teilung durch x^2+1. Für das Polynom x^3+3x^2+3x+1 zum Beispiel ist der Rest

$$\begin{array}{rl}
x^3+3x^2+3x+1 & : x^2+1 \;= x+3 \\
x^3 \quad\quad + \;\; x & \\
\quad 3x^2+2x+1 & \\
\quad 3x^2 \quad\quad +3 & \\
\quad\quad\quad 2x-2 &
\end{array}$$

Der Quotient ist $x+3$, der Rest $2x-2$.

Die Menge der Polynome $(x^2+1)\mathbb{Q}[x] \subset \mathbb{Q}[X]$ ist ein Ideal. Sie enthält alle Vielfachen von x^2+1, und Summen von solchen Vielfachen oder Produkte irgendeines Polynoms mit einem Vielfachen sind wiederum Vielfache von x^2+1. Also ist $\mathbb{Q}[x]/(x^2+1)\mathbb{Q}[x]$ ein Ring.

Der Divisionsalgorithmus stellt sicher, dass für jedes Polynom ein Rest mit Grad kleiner als 2 gefunden werden kann. Alle Reste sind also von der Form $a + bx$. Mit diesen Polynomen kann wie üblich gerechnet werden, nur $x \cdot x$ muss gesondert behandelt werden, da der Rest von x^2 bei Division durch $x^2 + 1$ gemäss folgender Rechnung

$$\begin{array}{ll} x^2 & : x^2 + 1 \ = 1 \\ x^2 + 1 & \\ \quad - 1 & \end{array}$$

-1 ist. Das spezielle Polynom $i = x$ in $\mathbb{Q}[x]/(x^2+1)\mathbb{Q}[x]$ verhält sich also genau so wie die imaginäre Einheit in $\mathbb{C}$, es ist also durchaus sinnvoll, dieses Element mit i zu bezeichnen, und für den Ring die Bezeichnung $\mathbb{Q}[i] = \mathbb{Q}[x]/(x^2 + 1)\mathbb{Q}[x]$ einzuführen.

Alternativ kann man $\mathbb{Q}[x]$ auch als eine Erweiterung des Ringes $\mathbb{Q}$ um ein neues Element x ansehen, welches die Gleichung $x^2 + 1 = 0$ erfüllt.

3.2.4. Bitvektoren. Bitvektoren sind spezielle Polynome, nämlich solche in $\mathbb{F}_2[X]$. Ein Polynom $a_0 + a_1X + a_2X^2 + \cdots + a_nX^n$ entspricht also einem Vektor von $n + 1$ bits $(a_0, \ldots, a_n)$, es kann mithin als ein Wort von $n + 1$ bit Länge geschrieben werden.

Multiplikationen von Bitvektoren sind deshalb von Interesse, weil sie Diffusion aufweisen. Werden zwei Bitvektoren multipliziert, so wirkt sich eine Einzelbit-Änderung in einem Faktor potentiell in so vielen Stellen des Resultates aus, wie der andere Faktor 1-Stellen hat. Um jedoch eine umkehrbare Operation zu erhalten, muss auch dividiert werden können. Im Allgemeinen ist dies sicher nicht möglich, wie auch die Division in $\mathbb{Z}$ im Allgemeinen nicht möglich ist. Hingegen ist in $\mathbb{Z}/n\mathbb{Z}$ die Division durch alle zu n teilerfremden Elemente möglich, es ist also auch zu hoffen, dass dies in einem geeigneten Restklassenring von $\mathbb{F}_2[x]$ ähnlich sein wird.

Operationen. In den voranstehenden Abschnitten wurde gezeigt, wie man Polynome addiert: die Koeffizienten werden addiert. Da die Addition in $\mathbb{F}_2$ gerade die XOR-Operation ist, läuft die Addition auf ein XOR der beiden $n + 1$-bit Wörter hinaus.

Die Multiplikation ist hingegen wesentlich komplexer, sie wird zu einer Faltung. Sie kann durch das bekannte Verfahren der schriftlichen Multiplikation (Ausmultiplizieren) realisiert werden, die unten beispielhaft

illustriert wird:

```
          1011001 * 110101011011001

                       110101011011001
                     110101011011001
                   110101011011001
                 110101011011001

                 11111001001111100001
```

Analog kann auch die schriftliche Division mit Rest für diese Polynome sehr einfach durchgeführt werden:

```
10110111100010100110 : 1010010011001 = 10010110
1010010011001
0001001101000010
   1010010011001
   001111101101101
     1010010011001
     01011111101001
      1010010011001
      00011011100000
```

Durch Multiplikation und Addition kann man die Korrektheit dieses Resultates überprüfen:

```
          10010110 * 1010010011001

                        1010010011001
                      1010010011001
                    1010010011001
                  1010010011001

                  10110111111001000110
Rest:                     11011100000

                  10110111100010100110
```

Für alle diese Operationen sind immer nur 1-bit-Abfragen, Shifts und XOR notwendig, sie lassen sich daher sehr effizient in Hardware realisieren, und auch reale Prozessoren haben keine Probleme mit diesen Operationen.

3.3. Zahlentheorie.

In diesem Abschnitt stellen wir einige Hilfsmittel der Zahlentheorie zusammen, die in den folgenden Kapiteln benötigt werden. Dabei legen wir besonderen Wert auf die Feststellung, dass die meisten Aussagen für ganze Zahlen auch eine Entsprechung in den Polynomring $K[x]$ haben. Dies erlaubt uns, neben der vertrauten Multiplikation auch die billigere (da ohne Übertrag) Multiplikation mit Bitvektoren als Quelle von Diffusion zu verwenden.

Dieser Abschnitt behandelt nur die elementaren Grundlagen, die in allen späteren Kapiteln von Nutzen sein werden. Speziellere Aussagen, wie sie zum Beispiel für die Verfahren von Diffie und Hellman oder Rivest, Shamir und Adleman (RSA) benötigt werden, werden bei der Behandlung dieser Verfahren erörtert.

3.3.1. Teilbarkeit. Die Eigenschaften eines Ringes genügen bereits, um den Begriff der Teilbarkeit zu definieren:

Definition 3.5. *In einem Ring R heisst das Element $a \in R$ durch $b \in R \setminus \{0\}$ teilbar, wenn es ein $q \in R$ gibt mit $a = bq$. Man schreibt $b|a$ falls b ein Teiler von a ist.*

Es gilt auch, dass a durch b teilbar ist, wenn a im Ideal bR enthalten ist: $a \in bR$. Im Ring $\mathbb{Z}$ läuft diese Definition auf den üblichen Begriff der Teilbarkeit hinaus. In $\mathbb{Q}$ ist jedes Element durch jedes von 0 verschiedene Element teilbar.

Im Polynomring $\mathbb{Z}[X]$ ist das Polynom X^2-1 durch $X-1$ und durch $X+1$ teilbar. Das Polynom X^2+1 hingegen ist nicht durch ein Polynom vom Grad 1 teilbar. Eine Faktorzerlegung müsste ja die Form $(X-a)(X-b) = X^2-(a+b)X+ab$ haben. Die ganzen Zahlen a und b müssten also die Eigenschaften $ab = 1$ und $a = -b$ haben. Daraus leitet sich die Gleichung $a^2 = -1$, aber die Quadrate von ganzen Zahlen sind immer ≥ 0. Im Ring $\mathbb{C}[X]$ der komplexen Polynome hingegen ist X^2+1 durch die Polynome $X+i$ und $X-i$ teilbar, ja es gilt $X^2+1 = (X+i)(X-i)$.

In $\mathbb{Z}$ und in den Polynomringen $K[X]$ lässt sich die Teilbarkeit mit Hilfe des Divisionsalgorithmus entscheiden, es steht also in jedem Fall ein praktikabler Algorithmus zur Verfügung.

3.3.2. Primzahlen. Eine besondere Rolle spielen Elemente, die keine "interessanten" Teiler haben. In Ringen, in denen jedes Ideal in der Form aR geschrieben werden kann, sind die von Primelementen p erzeugten Ideal pR *maximal*: es gibt keine Ideale, die echt zwischen pR und R liegen. Ein solches Ideal müsste von der Form mR sein, und ausserdem müsste m Vielfaches von p sein, also könnte $m = qp$ geschrieben werden. Daraus folgt jedoch, dass $mR = pqR \subset pR$ gilt, wegen $pR \subset mR$ folgt daraus $pR = mR$, das Ideal mR ist also gar nicht grösser.

Definition 3.6. *Zahl $p \in \mathbb{Z}$ heisst prim, wenn sie nur 1 und sich selbst als Teiler hat. Inbesondere folgt also für eine Primzahl p aus der Gleichung $ab = p$, dass entweder a oder b mit p identisch sein muss.*

Ein Element $p \in R$ heisst prim, wenn die einzigen Teiler die invertierbaren Elemente R^* von R oder Vielfache von p sind. Ist $p \in R$ prim, dann folgt aus $p = ab$, dass a oder b bis auf ein invertierbares Element mit p identisch sind, also $a = up$ oder $b = up$ mit $u \in R^*$.

Beweis: Ist $p \in \mathbb{Z}$ ein Primzahl, folgt aus $p|ab$ dass p mindestens eine der Zahlen a oder b teilen muss. Nennen wir den grössten gemeinsamen Teiler von b und p d. Dann gibt es Zahlen s, t mit $p = sd$ und $b = td$. Da p eine Primzahl ist bleiben nur die Möglichkeiten $d = p$ oder $d = 1$. Im ersten Fall ist $b = dp$, also ist p ein Teiler von b. Im zweiten Fall sind b und p teilerfremd, p ein Teiler von ab, also muss p ein Teiler von a sein.

Das Argument für $\mathbb{Z}$ ist weitgehend auch für einen beliebigen Ring R gültig. Für einen beliebigen Ring kann man jedoch nicht schliessen, dass $d = 1$ ist, sondern nur, dass d invertierbar ist.

Mit Hilfe vollständiger Induktion kann man die Aussage der Definition auf eine beliebige Zahl von Faktoren verallgemeinern, wir formulieren dies als Satz:

Satz 3.7. *Falls die Primzahl p das Produkt $n_1 \dots n_t$ teilt, teilt sie auch mindestens einen der Faktoren t_i.*

Falls ein Primelement $p \in R$ das Produkt $n_1 \dots n_t$ von Elementen $n_i \in R$ teilt, so teilt es auch mindestens einen der Faktoren.

Satz 3.8 (Primzahlsatz von Euklid). *Es gibt unendlich viele Primzahlen in $\mathbb{Z}$.*

Dieser Satz ist nicht verallgemeinerbar auf beliebige Ringe, in einem endlichen Ring R gäbe es gar nicht genug Elemente dafür.

Beweis: Angenommen, es gäbe nur endlich viele Primzahlen $p_1, \ldots, p_n$, so könnte man deren Produkt bilden, und eins dazu addieren:

$$N = 1 + p_1 \ldots p_n.$$

Bei Teilung durch eine der Primzahlen p_i ergibt sich immer der Rest 1, N ist also durch keine der Primzahlen teilbar. Ausserdem ist N grösser als alle Primzahlen, selbst aber keine. Folglich muss N durch mindestens eine Primzahl teilbar sein. Dieser Widerspruch zeigt, dass die Annahme, es gäbe nur endlich viel Primzahlen, unhaltbar ist.

3.3.3. Primfaktorzerlegung. Zwei Primelemente eines Ringes R können wir als gleich betrachten, wenn sie sich nur um einen Faktor unterscheiden, der in R invertierbar ist. Die Menge der invertierbaren Elemente heisst Einheitengruppe und wird mit R^* bezeichnet. In $\mathbb{Z}$ sind 1 und -1 die einzigen invertierbaren Elemente. In einem Körper sind alle Elemente ausser 0 invertierbar, $K^* = K \setminus \{0\}$.

Ist $u \in R^*$ eine Einheit, ist $p \in R$ genau dann ein Primelement, wenn up ein Primelement ist. In $\mathbb{Z}$ bedeutet dies, dass uns das Vorzeichen einer Primzahl nicht weiter interessiert.

Die Menge der Primelemente $P(R)$ eines Ringes ändert sich also nicht, wenn man mit Elementen von R^* multipliziert. Ausserdem wird $P(R)$ in Klassen von Primelementen eingeteilt, die sich um einen Faktor aus R^* unterscheiden.

Bespiel: Primelemente in $\mathbb{Z}$. Die Primelemente von $\mathbb{Z}$ sind

$$P(\mathbb{Z}) = \{\pm 2, \pm 3, \pm 5, \pm 7, \pm 11, \ldots\}$$

durch die Einheitengruppe $\{\pm 1\}$ werden jeweils entgegengesetzte Primelemente zusammengefasst, also entsteht die Einteilung

$$\{\{\pm 2\}, \{\pm 3\}, \{\pm 5\}, \{\pm 7\}, \{\pm 11\}, \ldots\}.$$

Beispiel: Primelemente in $\mathbb{Q}[x]$. In den Polynome mit rationalen Koeffizienten $\mathbb{Q}[x]$ sind die Einheiten gerade die konstanten, von 0 verschiedenen Polynome, $\mathbb{Q}[x]^* = \mathbb{Q}^*$. Die Primelemente können also mit einer nicht verschwindenden Zahl multipliziert werden. Dies ermöglich es, jedes Primelement so mit einem Faktor zu multiplizieren, dass der Koeffizient des Terms mit dem höchsten Grad 1 ist. Wir können also Polynomen der Form

$$x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$

ausgehen. Man nennt solche Polynome *normiert*.

Alle Polynome der Form $x-a$ sind Primpolynome. Es gibt also bereits zum Grad 1 unendlich viele verschiedene Primpolynome. Zum Grad 2 müssen wir Primpolynome in der Form $x^2 + px + q$ suchen. Aus der Analysis ist bekannt, dass sich solche Polynome genau dann in Faktoren der Form $x - a$ zerlegen lassen, wenn $x^2 + px + q$ rationale Nullstellen hat. Nach der Lösungsformel für die quadratische Gleichung

$$x_{1,2} = -\frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q}$$

ist das genau dann möglich, wenn

$$\sqrt{\frac{p^2}{4} - q} \in \mathbb{Q}$$

ist. Andernfalls ist $x^2 + px + q$ ein Primelement.

Man beachte, dass in $\mathbb{C}[x]$ die Polynome $x - a$ die einzigen Primpolynome sind. Über $\mathbb{C}$ lässt sich jedes Polynom in lineare Faktoren zerlegen, jedes Polynom über $\mathbb{C}$ hat alle seine Nullstellen in $\mathbb{C}$.

Der Divisionsalgorithmus in $\mathbb{Z}$ stellt sicher, dass zu zwei Zahlen a und b immer ein eindeutiger Quotient q und ein Rest $0 \leq r < b$ existiert mit $a = qb + r$. Für die Polynome ist dies nicht so einfach, weil sie nicht angeordnet werden können. Immerhin kann man sagen, dass zu zwei Polynomen a und b ein Quotientenpolynom q existiert sowie ein Rest r, dessen Grad kleiner ist als der Grad von b:

$$a = bq + r, \qquad \deg(r) < \deg(b).$$

Dies genügt jedoch bereits, um die Existenz einer Primfaktorzerlegung zu beweisen:

Satz 3.9 (Primfaktorzerlegnung). *Ist $n \in R$, wobei $R = \mathbb{Z}$ oder R ein Polynomring mit Koeffizienten in einem Körper, dann gibt es bis auf Einheiten in R eindeutig bestimmte Primelemente $p_i \in R$ und Exponenten $\alpha_i \in \mathbb{N}$ derart, dass*

$$n = \varepsilon p_1^{\alpha_1} \dots p_l^{\alpha_l},$$

wobei $\varepsilon \in R^*$ eine Einheit ist.

In der vertrauten Situation $R = \mathbb{Z}$ bedeutet dies, dass die Primfaktorzerlegung bis auf die Vorzeichen der Faktoren eindeutig ist.

Beweis: Zunächst beweisen wir die Existenz. Falls n selbst kein Primelement ist, ist es durch $b \in R$ teilbar, kann also in der Form $n = bq$ geschrieben werden, wobei $\deg(q) < n$ ist. Wiederholt man dieses Argument für q, kann man nacheinander immer neue Faktoren abspalten, der verbleibende Faktor hat dabei immer kleineren Grad. Daher muss das Verfahren in endlich vielen Schritten terminieren, und ergibt eine Faktorzerlegung von n.

3.4. Der Euklidische Algorithmus.

Gerade weil die Division oft nicht durchführbar ist, wird die Frage, für welche Paare (a, b) der Quotient dennoch berechnet werden kann, interessant. Besonders interessant sind auch jene Elemente b, die invertiert werden können. In diesem Abschnitt soll mit dem euklidischen Algorithmus eine praktische Methode aufgestellt werden, mit der multiplikative Inverse in vielen der Ringe R/I berechnet werden können.

3.4.1. Nullteiler. Leider gibt es Hindernisse grundsätzlicher Art gegen das Invertieren, sogenannte Nullteiler führen dazu, dass keine sinnvolle Inverse definiert werden kann.

Definition 3.10. *Ein Ring R heisst nullteilerfrei, wenn aus $ab = 0$ für $a, b \in R$ immer folgt, dass mindestens einer der Faktoren verschwindet, also $a = 0$ oder $b = 0$.*

In einem Ring mit Nullteilern ist es nicht mehr möglich, einen eindeutigen Quotienten zu definieren. In der Menge der Reste bei Teilung durch 6 gilt zum Beispiel $2 \cdot 3 = 0$, aber auch $0 \cdot 3 = 0$. Jeder der Reste 2 und 0 ist also ein Kandidat für den Quotienten $0/3$. Damit ist aber

auch jeder andere Quotien durch 3 nicht mehr eindeutig, denn ist ist $a/3 = (a+0)/3 = a/3 + 0/3$. Wenn $0/3$ nicht eindeutig ist, kann auch $a/3$ nicht eindeutig sein.

Somit kann eine multiplikative Inverse nur dann sinnvoll sein, wenn der Divisor (der Nenner) kein Nullteiler ist.

3.4.2. Grad. Beim klassischen Divisionsalgorithmus kann bewiesen werden, dass er in endlich vielen Schritten terminiert, weil die Zwischenresultate immer kleiner werden, also irgen wann 0 erreichen müssen, womit die Division auch abgeschlossen ist.

Bei Polynomen ist gibt es keinen unmittelbaren Vergleich. Trotzdem kann man beweisen, dass die Polynomdivision terminiert, indem man bemerkt, dass in jedem Schritt der Grad des Polynoms abnimmt, aber nicht kleiner als 0 werden kann. Wir formalisieren die Eigenschaft des Grades in der nachstehenden Definition.

Definition 3.11. *Der Grad des Polynoms $p(X) \in R[X]$ ist die grösste Zahl k, für die in der Darstellung*

$$p(X) = a_n X^n + \cdots + a_1 X + a_0$$

der a_k von Null verschieden ist. Für das Polynom $p(X) = 0$ setzt man $\deg(p) = -\infty$. Der Koeffizient a_k heisst der Leitkoeffizient.

Ein Ring R mit einer Abbildung mit den gleichen Eigenschaften wie dem Grad heisst ein euklidischer Ring. Die meisten der unten über $\mathbb{Z}$ und über Polynome abgeleiteten Sätze gelten auch in euklidischen Ringen.

Die Eigenschaft, nullteilerfrei zu sein, überträgt sich auf den Polynomring. Das Produkt zweier Polynome mit Leitkoeffizienten a_n und b_m enthält für den Term X^{n+m} genau den Koeffizienten $a_n b_m$. Da R keine Nullteiler enthält, und beide Faktoren von 0 verschieden sind, kann $a_n b_m$ nicht 0 sein. Damit haben wir bewiesen:

Lemma 3.12. *Falls R keine Nullteiler hat, gilt für zwei Polynome $p_1, p_2 \in R[X]$ gilt $\deg(p_1 p_2) = \deg(p_1) + \deg(p_2)$.*

3.4.3. Grösster gemeinsamer Teiler. In Abschnitt 3.3.1. haben wir allgemeine Definition der Teilbarkeit gegeben. Die Menge der durch a teilbaren Elemente hat sich als Ideal herausgestellt in den Beispielen war dies aR, dasselbe gilt natürlich für die durch b teilbaren Elemente.

Die Menge der durch a und b teilbaren Elemente ist daher $aR + bR$, und muss ebenfalls ein Ideal sein. Falls in R alle Ideale in der Form gR geschrieben werden können, können wir bereits schliessen, dass $g = as + bt$ für geeignete Elemente $s, t \in R$. Das Element g muss ein Teiler von a und b sein, ausserdem ist jeder andere Teiler von a und b auch ein Teiler von g, in diesem Sinne ist g der grösste aller gemeinsamen Teiler von a und b:

Definition 3.13. *$t \in R$ heisst ein gemeinsamer Teiler von $a \in R$ und $b \in R$, falls $t|a$ und $t|b$. t heisst grösster gemeinsamer Teiler von a und b, falls folgende Bedingungen erfüllt sind:*

(1) *t ist eine Teiler von a und b, und*
(2) *jeder andere Teiler von a und b teilt auch t.*

So sind sowohl 2 als auch 3 gemeinsame Teiler von 12 und 18, aber erst 6 ist der grösste gemeinsame Teiler.

Satz 3.14 (Euklidischer Algorithmus). *Falls die ganze Zahl $d \in \mathbb{Z}$ der grösste gemeinsame Teiler von $a, b \in \mathbb{Z}$ ist, gibt es Zahlen $s, t \in \mathbb{Z}$ mit $d = as + bt$.*

Die oben angegebene Beweisskizze mit Hilfe von Idealen setzt einige Kenntnisse der Ringtheorie voraus, die uns nicht zur Verfügung stehen. Ausserdem beschränkt sie sich darauf, die Existenz nachzuweisen, während uns mit Blick auf die Anwendung auch die praktische Berechnung von s und t interessiert. Der folgende Beweis des Satzes 3.14 benötigt die Division mit Rest, die wir wie folgt formulieren können:

Satz 3.15. *Zu zwei Zahlen $a, b \in \mathbb{Z}, b \neq 0$ gibt es $q, r \in \mathbb{Z}$ mit $a = qb + r$ mit $r < b$. Zu zwei Polynomen $a, b \in K[X]$ gibt es Polynome $q, r \in K[X]$ mit $a = qb + r$ mit* $\deg(r) < \deg(b)$.

Mit dem üblichen Divisionsalgorithmus haben wir den Beweis, dass solche Zahlen s und t immer konstruiert werden können.

Beweis: Zunächst stellen wir fest, dass der grösste gemeinsame Teiler d von a und b auch ein Teiler von $a - bq_1 = r_1$ ist. Da $r_1 < b$ ist, können wir erneut den Divisionsalgorithmus auf b und r_1 anwenden, um Zahlen q_2 und r_2 zu finden mit $b = q_2 r_1 + r_2$. Wieder ist $r_2 < r_1$. Somit lässt sich

eine abnehmende Folge von Zahlen r_i und zugehörigen q_i konstruieren, so dass d ein Teiler aller r_i ist:

$$\begin{aligned} a &= bq_1 + r_1 \\ b &= r_1q_2 + r_2 \\ r_1 &= r_2q_3 + r_3 \\ r_2 &= r_3q_4 + r_4 \\ &\vdots \\ r_{n-4} &= r_{n-3}q_{n-2} + r_{n-2} \\ r_{n-3} &= r_{n-2}q_{n-1} + r_{n-1} \\ r_{n-2} &= r_{n-1}q_n \end{aligned}$$

Nach endlich vielen Schritten, sagen wir bei $i = n$, ist $r_n = 0$. r_{n-1} ist also der kleinste all dieser Zahlen, die von d geteilt werden. Wir behaupten, dass sogar $d = r_{n-1}$ ist. Dazu genügt es zu zeigen, dass r_{n-1} ein Teiler von a und b ist. Wegen der der letzten Gleichung ist $r_{n-1}|r_{n-2}$. Die zweitletzte zeigt, dass $r_{n-1}|r_{n-3}$, usw. Wir schliessen, dass r_{n-1} ein Teiler von a und b ist. Da r_{n-1} ausserdem den grössten gemeinsamen Teiler teilt, muss r_{n-1} der grösste gemeinsame Teiler sein.

Aus den obigen Gleichungen lässt sich jetzt auch die Beziehung $as + bt = d$ ableiten. Die zweiteletzte Gleichung sagt ja, dass $r_{n-3} - r_{n-2}q_{n-1} = d$. Darin kann man r_{n-2} ersetzen durch $r_{n-4} - r_{n-3}q_{n-2}$, man erhält:

$$d = r_{n-3}(1 - q_{n-2}) + q_{n-1}r_{n-4}.$$

Durch Anwendung aller Gleichungen in umgekehrter Reihenfolge wird man zurückgeführt auf einen Ausdruck der gesuchten Form.

Ersetzt man im soeben für $\mathbb{Z}$ geführten Beweis "ganze Zahlen" durch "Polynome" und Vergleiche durch Vergleiche des Grades der Polynome erhält man den Beweis des euklidischen Algorithmus für Polynome.

Für die praktische Rechnung ist dieses Vorgehen unpraktisch, weil man alle Zwischenresultat speichern muss, um sie am Ende wieder zu den Werten s und t zusammensetzen zu können.

3.4.4. Praktische Durchführung. Verfolgt man die Ereignisse bei der Durchführung des euklidischen Algorithmus im vorangehenden Abschnitt, kann man s und t auch durch laufende Verwertung der anfallenden Quotienten und Reste ermitteln. Eine elegante Beschreibung davon findet man in [1], wir reproduzieren sie ohne Beweis.

Algorithmus 3.16 (grösster gemeinsamer Teiler). *Gegeben sind zwei Elemente $u, v \in R$, wobei der Divisionsalgorithmus im Ring R eindeutig durchführbar sein soll. Dann findet man mit dem folgenden Algorithmus drei Elemente $(u_1, u_2, u_3) \in R^3$ mit der Eigenschaft*

$$uu_1 + vu_2 = u_3 = \mathrm{ggT}(u, v).$$

(i) *Initialisiere die Vektoren $\vec{u} = (1, 0, u)$ und $\vec{v} = (0, 1, v)$.*
(ii) *Sofern $v_3 = 0$ ist, ist der Algorithmus beendet.*
(iii) *Berechne den Quotienten q von u_3/v_3, und setze*

$$\begin{aligned} \vec{t} &:= \vec{u} - q\vec{v} \\ \vec{u} &:= \vec{v} \\ \vec{v} &:= \vec{t} \end{aligned}$$

Es ist klar, dass nach Schritt (iii) an der dritten Stelle im Vektor $\vec{v}$ der Rest der Division von u_3 durch v_3 steht.[3]

Beispiele. Wir führen den Algorithmus an zwei Beispielen durch, einem für ganze Zahlen, und einem für Polynome mit rationalen Koeffizienten.

1. Man finde den grössten gemeinsamen Teiler von 40902 und 24140. Die Rechnung gemäss dem Algorithmus kann in der folgenden Tabelle wiedergegeben werden:

[3] In der Vorlesung wurde dieser Algorithmus leicht anders formuliert, u_1 und u_2 hiessen dort s und s', v_1 und v_2 hiessen t und t'. v_3 war jeweils der Rest der Division u_3 durch v_3. In der tabellarischen Darstellung war die Reihenfolge der Spalten jeweils anders, was jedoch auf die Rechnung ohne Einfluss ist.

q	u_1	u_2	u_3	v_1	v_2	v_3
	1	0	40902	0	1	24140
1	0	1	24140	1	−1	16762
1	1	−1	16762	−1	2	7378
2	−1	2	7378	3	−5	2006
3	3	−5	2006	−10	17	1360
1	−10	17	1360	13	−22	646
2	13	−22	646	−36	61	68
9	−36	61	68	337	−571	34
2	337	−571	34	−710	1203	0

Der grösste gemeinsame Teiler ist also 34 und es gilt

$$337 \cdot 40902 - 571 \cdot 24140 = 34,$$

wie man durch nachrechnen bestätigt.

2. Bestimme den grössten gemeinsamen Teiler der Polynome $2x^2 + 4x + 2$ und $x^2 - 1$.

q	u_1	u_2	u_3	v_1	v_2	v_3
	1	0	$2x^2 + 4x + 2$	0	1	$x^2 - 1$
2	0	1	$x^2 - 1$	1	-2	$4x + 4$
$\frac{1}{4}(x-1)$	1	-2	$4x + 4$	$-\frac{1}{4}(x-1)$	$\frac{1}{2}x + \frac{3}{2}$	0

Der grösste gemeinsame Teiler ist also $4(x + 1)$ und es gilt

$$1 \cdot (2x^2 + 4x + 2) + (-2) \cdot (x^2 - 1) = 4x + 4.$$

3.4.5. Anwendungen des grössten gemeinsamen Teilers.

Definition 3.17. *Zwei Elemente a und b von R heissen teilerfremd, wenn ihr grösster gemeinsamer Teiler 1 ist.*

Folgerung 3.18. *Zwei ganze Zahlen a und b sind genau dann teilerfremd, wenn es Zahlen a' und b' gibt mit $aa' + bb' = 1$.*

Lemma 3.19. *Sind a, b teilerfremd, und ist $a|bc$, dann gilt $a|c$.*

Beweis: Da a, b teilerfremd sind, gibt es $x, y \in \mathbb{Z}$ mit $ax + by = 1$, oder $c = axc + byc$. Nun ist aber $a|bc$, also auch $a|(acx + bcy) = c$.

Das Intersse an teilerfremden Elementen rührt von der teilweisen Invertierbarkeit her. Sind die Zahlen $a, b \in \mathbb{Z}$ teilerfremd, dann gibt es Zahlen $s, t \in \mathbb{Z}$ mit $as + bt = 1$. Betrachtet man Reste bezüglich b, also den Ring $\mathbb{Z}/b\mathbb{Z}$, folgt $as = 1 \in \mathbb{Z}/b\mathbb{Z}$, a ist also in $\mathbb{Z}/b\mathbb{Z}$ invertierbar, die Inverse kann mit dem euklidischen Algorithmus berechnet werden.

Ist b eine Primzahl, dann ist jede Zahl $0 < a < p$ teilerfremd zu p, also gibt es nach dem eben gesagten ein Element $s \in \mathbb{F}_p$ mit $as = 1 \in \mathbb{F}_p$.

Beispiele.

1. Wir berechnen die Inverse von $47 \in \mathbb{F}_{1291}$. Dazu müssen wir den euklidischen Algorithmus für 47 und 1291 durchführen:

q	u_1	u_2	u_3	v_1	v_2	v_3
	1	0	1291	0	1	47
27	0	1	47	1	−27	22
2	1	−27	22	−2	55	3
7	−2	55	3	15	−412	1
3	15	−412	1	−47	2939	0

   Die Kontrolle ergibt $15 \cdot 1291 - 412 \cdot 47 = 1$. Die Restklasse von -412 ist $1291 - 412 = 879$, daher gilt in $\mathbb{F}_{1291}$ die Relation $47 \cdot 879 = 1$.
2. Der Algorithmus zur Bestimmung der Inversen funktioniert auch im Ring der Polynome $\mathbb{Q}[x]/(x^2+1)\mathbb{Q}[x]$. Da $x^2 + 1$ in $\mathbb{Q}[x]$ ein Primpolynom ist, ist es teilerfremd zu $x + 1$, wir bestimmen die Inverse von $x + 1$ in $\mathbb{Q}[x]/(x^2+1)\mathbb{Q}[x]$.

   Der euklidische Algorithmus für $x^2 + 1$ und $x + 1$ ergibt:

q	u_1	u_2	u_3	v_1	v_2	v_3
	1	0	x^2+1	0	1	$x+1$
$x-1$	0	1	$x+1$	1	$-x+1$	2
$\frac{1}{2}(x+1)$	1	$-x+1$	2	$-\frac{1}{2}(x+1)$	$\frac{1}{2}(3-x^2)$	0

   Durch nachrechnen finden wir, dass

$$1 \cdot (x^2 + 1) + (-x + 1) \cdot (x + 1) = 2.$$

Für die Inverse brauchen wir natürlich 1 auf der rechten Seite, was aber nicht weiter problematisch ist, da wir in $\mathbb{Q}[x]$ ja durch 2 dividieren können: also

$$\frac{1}{2} \cdot (x^2 + 1) + \frac{-x + 1}{2} \cdot (x + 1) = 1.$$

Die Inverse von $x + 1$ in $\mathbb{Q}[x]/(x^2 + 1)\mathbb{Q}[x]$ ist also $\frac{1}{2}(1 - x)$. Tatsächlich ist

$$\begin{aligned}(x + 1) \cdot \frac{1}{2}(-x + 1) &= -\frac{x^2 - 1}{2} \\ &= -\frac{x^2 + 1 - 2}{2} \\ &= 1 \in \mathbb{Q}[x]/(x^2 + 1)\mathbb{Q}[x].\end{aligned}$$

Kapitel 4. Symmetrische Verschlüsselungsverfahren

Symmetrische Verfahren verwenden für Ver- und Entschlüsselung den gleichen Algorithmus und den gleichen Schlüssel. Es ist nicht möglich, das Verschlüsselungsverfahren zu veröffentlichen, ohne die Entschlüsselung ebenfalls preiszugeben. So können symmetrische Verfahren nur dann eingesetzt werden, wenn Schlüssel vorgängig auf einem unabhängigen, sicheren Kanal ausgehandelt worden sind.

Wir untersuchen in diesem Kapitel die symmetrischen Verfahren auch im Hinblick auf die Frage, wie die Diffusion verbessert werden kann. Dieses Problem stellt sich auch bei Hash-Funktionen, bei denen die Umkehrbarkeit nicht gefordert ist.

Die symmetrischen Verfahren zeichnen sich durch wesentlich grösseren Durchsatz aus, als die zahlentheoretischen asymmetrischen Verfahren. Deshalb werden in der Praxis immer verschiedene Verfahren gemischt, zum Beispiel ein asymmetrisches Verfahren (Diffie-Hellman oder RSA) zur Aushandlung eines Schlüssels, der anschliessend im Rahmen eines symmetrischen Verfahrens eingesetzt wird.

4.1. Hashfunktionen.

Kryptographische Hashfunktionen oder Einweg-Funktionen produzieren aus einem grösseren (oder kleineren) Klartext P ein kurzes Chiffrat, den Hashcode $C = h(P)$. Eine gute Hashfunktion erfüllt folgende Rahmenbedingungen:

1. Die Änderung eines einzelnen Klartextbits ändert im Mittel genau 50% der Chiffratbits. Jede Abweichung von diesem Mittelwert kann zur Kryptanalyse des Hashalgorithmus verwendet werden.
2. Es ist praktisch nicht möglich, zu einem gegebenen Klartext P einen zweiten Klartext P' zu produzieren, der den gleichen Hashwert hätte: $h(P) = h(P')$.
3. Es ist praktisch nicht möglich, zwei Klartexte P_1 und P_2 mit dem gleichen Hashwert $h(P_1) = h(P_2)$ zu produzieren. Dies ist etwas leichter als die vorangehende Aufgabe, einen Text mit einem vorgegeben Hashwert zu produzieren.

Selbstverständlich sollte die Funktion h effizient berechnet werden, vorzugsweise auf einer breiten Palette von Prozessoren. Nachfolgend wer-

den MD2 und MD5 als Beispiele von Algorithmen vorgestellt, welche auf 8bit- bzw. 32bit-Prozessoren optimiert sind. Der letze Abschnitt über HMAC zeigt, wie aus einer gegebenen Hashfunktion und einem Schlüssel eine neue Hashfunktion konstruiert werden kann. Es entsteht also eine sehr grosse Familie verschiedener Hashfuntkionen, die von einem Schlüssel abhängig sind.

4.1.1. MD2. MD2 ist auf 8bit-Prozessoren optimiert, die keine grösseren Einheiten als Bytes verarbeiten können. Der Algorithmus arbeitet mit Blocks von 16 Bytes Länge, um eine eindeutige Ausgangslage zu schaffen, muss der Klartext durch Anhängen von i Bytes vom Wert i, $1 \leq i \leq 16$ auf ein Vielfaches von 16 verlängert werden. Das Padding muss auch angewendet werden, wenn der Klartext bereits ein Vielfaches von 16 lang ist. Andernfalls können triviale Klartexte konstruiert werden, die alle den gleichen Hashwert haben. Ohne Padding eines vollen 16-Byte Blockes würden die Klartexte mit den letzten partiellen 16bit-Blocks

```
xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx 01
xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
```

auf den gleichen letzen Block

```
xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx 01
```

verlängert. MD2 verlangt ausserdem, dass eine 16 Byte Prüfsumme angehängt wird.

Der Algorithmus verwendet eine Permutation $\sigma \in S_{256}$ der Zahlen $0 \ldots 255$. Der genaue Wert dieser Permutation ist nicht von Bedeutung, um aber jegliche Willkür auszuschliessen, können die Werte $\sigma(i)$ aus der Dezimalentwicklung von π abgeleitet werden. Die Permutation σ kann als eine Tabelle von 256 Bytes realisiert werden, also als

```
char    S[256];
```

MD2 baut den Hashcode in einem Zustandsspeicher von 48 Byte Länge auf, den wir mit $X_0, \ldots, X_{47}$ notieren. Zu Beginn wird dieser Zustand mit 0-Bytes gefüllt.

Für jeden 16-Byte-Block das Klartextes wird jetzt die folgende Funktion berechnet:

1. Der Datenblock wird in die Bytes $X_{16}, \ldots, X_{31}$ kopiert.
2. Die letzten 16 Bytes des Zustandes werden durch XOR aus den vorangehenden zwei Blocks bestimmt:

$$X_i = X_{i-16} \oplus X_{i-32}, \qquad 32 \leq i < 48$$

.

3. Auf den Zustandsblock wird die folgende in C-Code notierte Kompressionsfunktion angewendet

```
t = 0;
for (j = 0; j < 18; j++) {
        for (k = 0; k < 48; k++) {
                t = X[k] ^ S[t];
                X[k] = t;
                t = (t +  j) % 256;
        }
}
```

Der Hashwert ist der Inhalt der ersten 16 Bytes des Zustandsvektors, also

```
X[0],...,X[15]
```

Die innere Schleife wird nicht weniger als 864 mal ausgeführt. Jedes Byte im Zustandsarray wird mindestens 18 mal verwendet, um einen neuen `t`-Wert zu bestimmen. Ausserdem fliesst der Wert auch in die Berchnung des neuen `t` ein, welches wiederum bestimmt, welcher Wert der Permutation im nächsten verwendet werden soll.

Eine 1-bit Änderung im ersten Byte des Zustandsarrays `X[0]` wirkt sich also schon im ersten Schritt auf `t` aus. Im zweiten Schritt wird das Byte `X[1]` neu berechnet, indem zum bisherigen `X[1]` der Wert $\sigma(t)$ mit XOR hinzuaddiert wird. Somit wird sich `X[1]` ebenfalls ändern, etc.

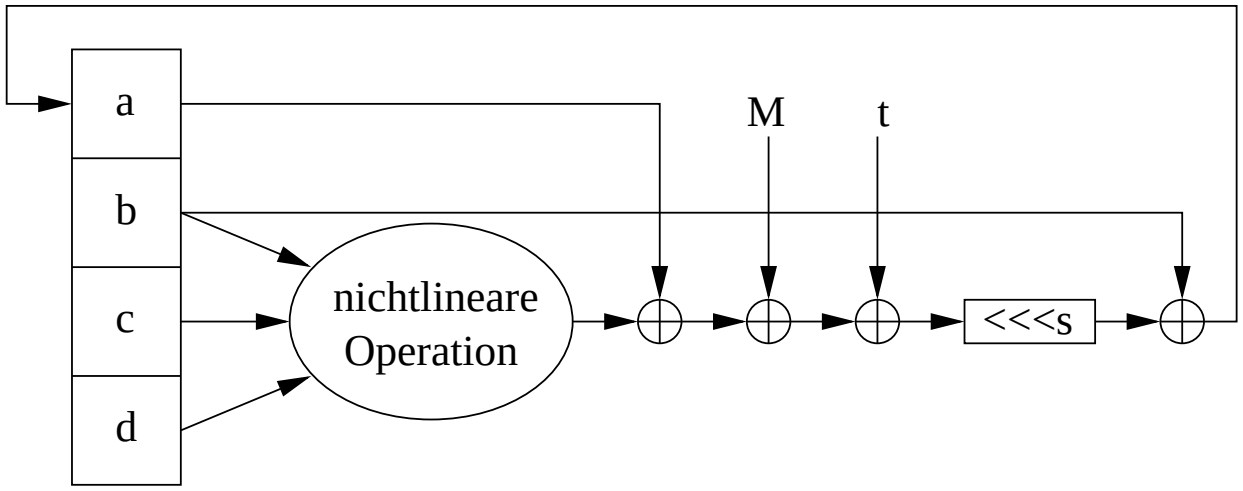

Abbildung 4.1. MD5 Operation.

Der MD2-Algorithmus erhält die Diffusion also daher, dass mit Hilfe der `t`-Variablen Werte in jedem Byte in die nächste Runde übertragen werden, und damit auch die nachfolgenden Bytes beeinflussen.

Wegen der vielfachen Wiederholung der inneren Schleife ist MD2 vergleichsweise langsam, ausserdem kann er von den Fähigkeiten von 32bit-Prozessoren nicht profitieren, da die Operationen nicht parallelisierbar sind. Ein neuer Schleifendurchgang ist erst möglich, wenn das neue `t` berechnet ist, welches in jeder Zeile der inneren Schleife benötigt wird.

4.1.2. MD5. MD5 liefert ebenfalls einen 128 Bit langen Hashwert, arbeitet aber durchgängig mit 32bit-Einheiten, so dass er auf 32bit-Prozessoren bedeutend effizienter ist als MD2.

MD5 arbeitet immer mit 64 Byte (512 bit) langen Blöcken, der Klartext muss also wie bei MD2 zunächst aufgefüllt werden. Dazu wird ein einzelnes 1-bit gefolgt von so vielen 0-bits angehängt, dass die Länge des resultierenden Textes genau 64 weniger ist als ein Vielfaches von 512 bit. In die letzten 64 bit wird die 64-bit Darstellung der Länge des Klartextes vor dem Anhängen von Pad-Bits eingefüllt.

Der Zustand von MD5 besteht aus 128bit, die als vier 32bit Variablen A, B, C und D betrachtet werden. Diese werden mit den folgenden Daten initialisiert:

```
A        B        C        D
01234567 89abcdef fedcba98 76543210
```

Nun kann der Algorithmus beginnen. Für jeden 512bit-Block des Klartextes werden die folgenden Schritte ausgeführt:

1. Die Werte der Subblocks A, B, C und D werden in Variablen a, b, c und d zwischengespeichert.
2. Auf a, b, c und d werden vier Runden angewendet. Jede Runde besteht aus 16 Operationen folgender Struktur, die auch in Abbildung 4.1. illustriert ist:
   (i) Drei der Variablen werden mit einer nichtlinearen Operation verknüpft. Die Lineare Operation ist in jeder Runde konstant, die übrigen Parameter ändern.
   (ii) Das Resultat wird zur vierten Variablen mit XOR addiert.
   (iii) Ein 32-bit Subblock des aktuellen Klartextblockes wird hinzuaddiert. In der ersten Runde werden die Klartextblocks in ihrer natürlichen Reihenfolge verwendet, in späteren Runden jedoch in einer Permutation.
   (iv) Eine in jeder Operation andere Konstante t_j wird hinzuaddiert. t_j ist der ganzzahlige Anteil von $2^{32} \sin j$, ist also nicht willkürlich.
   (v) Die Bits des Resultates werden zyklisch um eine wechselnde Zahl von Bits vertauscht.
   (vi) Einer der drei im ersten Schritt verabeiteten Werte wird zum Resultat hinzuaddiert.
   (vii) Das Resultat ersetzt den im zweiten Schritt verwendeten Subblock.
3. Das Resultat der Rundenoperationen wird zum Inhalt von A, B, C und D mit XOR hinzuaddiert.

Der Hashwert ist der Inhalt von A, B, C und D.

Die nichtlinearen Operationen von drei 32bit breiten Variablen, die zum Einsatz kommen, sind:

$$
\begin{aligned}
F(X,Y,Z) &= (X \wedge Y) \vee ((\neg X) \wedge Z) \\
G(X,Y,Z) &= (X \wedge Y) \vee (Y \wedge (\neg Z)) \\
H(X,Y,Z) &= X \oplus Y \oplus Z \\
I(X,Y,Z) &= Y \oplus (X \vee (\neg Z))
\end{aligned}
$$

Die erste Runde verwendet F, die zweite G etc. Die einzelnen Schritte der ersten Runde verwenden die abgekürzt als $FF(a, b, c, d, M, s, t)$ geschriebene Operation, die a ersetzen durch

$$b \oplus ((a \oplus F(b, c, d) \oplus M \oplus t) <<< s).$$

Dabei ist M einer der Klartextblöcke (32bit), und t eine der Konstanten t_j. Die vollständige erste Runde ist zum Beispiel:

$$\begin{aligned}
&FF(a, b, c, d,\ M_0,\ 7,\texttt{0xd76aa478})\\
&FF(d, a, b, c,\ M_1,12,\texttt{0xe8c7b756})\\
&FF(c, d, a, b,\ M_2,17,\texttt{0x242070db})\\
&FF(b, c, d, a,\ M_3,22,\texttt{0xc1bdceee})\\
&FF(a, b, c, d,\ M_4,\ 7,\texttt{0xf57c0faf})\\
&FF(d, a, b, c,\ M_5,12,\texttt{0x4787c62a})\\
&FF(c, d, a, b,\ M_6,17,\texttt{0xa8304613})\\
&FF(b, c, d, a,\ M_7,22,\texttt{0xfd469501})\\
&FF(a, b, c, d,\ M_8,\ 7,\texttt{0x698098d8})\\
&FF(d, a, b, c,\ M_9,12,\texttt{0x8b44f7af})\\
&FF(c, d, a, b,M_{10},17,\texttt{0xffff5bb1})\\
&FF(b, c, d, a,M_{11},22,\texttt{0x895cd7be})\\
&FF(a, b, c, d,M_{12},\ 7,\texttt{0x6b901122})\\
&FF(d, a, b, c,M_{13},12,\texttt{0xfd987193})\\
&FF(c, d, a, b,M_{14},17,\texttt{0xa679438e})\\
&FF(b, c, d, a,M_{15},22,\texttt{0x49b40821})
\end{aligned}$$

Wir verzichten auf eine vollständige Auflistung aller Runden, da mit den bisherigen Bespielen bereits eindrücklich gezeigt wurde, wie MD5 aufgebaut ist.

Aus der Konstruktion lässt sich auch verfolgen, wie die Diffusion entsteht. Durch die stets wechselnden Shifts (7, 12, 17, 22 in der ersten Runde, in den nachfolgenden Runden werden andere Werte verwendet) wirken sich Einzelbitänderungen in einer Runde bereits auf jedes einzelne Byte des Zustandsspeichers aus. Insgesamt werden die Shifts 4–7,

9–12, 14–17 und 20–23 verwendet. Durch Kombination können sich Einflüsse aber auch um Distanzen $31 = 11 + 20$ (ein Bit nach rechts) und $33 = 12 + 21$ (ein Bit nach links) verbreiten, die Shifts sind also offensichtlich so gewählt, dass jedes Bit des Hashcodes von jedem Input-Bit abhängt.

Während auf einem Pentium MMX 266 MHz 972kB pro Sekunde mit MD2 verarbeitet werden können, schafft der gleiche Prozessor 44MB mit MD5, MD5 ist also mindestens 45 mal schneller.

4.1.3. HMAC. Gemäss RFC2104 kann aus einer beliebigen Hashfunktion h eine neue Hashfunktion HMAC konstruiert werden, die von h und einem Schlüssel abhängt. Eine solche Funktion kann von zwei Partnern verwendet werden, die bereits über einen gemeinsamen geheimen Schlüssel verfügen, um die Integrität und Authentizität von Meldungen ohne die aufwendigen asymmetrischen Verfahren nachzuweisen.

HMAC arbeitet mit Blöcken von B Bytes Länge. Der Einfachheit halber gehen wir daher davon aus, dass der Schlüssel k ebenfalls diese Länge hat.

Definition 4.1 (HMAC). *Die Hashfunktion* $\mathrm{HMAC}_{h,k}$ *ist*

$$\mathrm{HMAC}_{h,k}(m) = h(k \oplus o, h(k \oplus i, m)),$$

darin sind i ein String von B Kopien von `0x36` *und o ein String von B Kopien von* `0x5c`.

4.2. DES.

Im vorangehenden Abschnitt über Hashfunktionen haben wir gesehen, wie durch Zusammensetzen von elementaren Operationen wie XOR, Shifts und Permutationen aus einem Klartext ein Chiffrat entstehen kann, bei dem jedes Bit von allen Bits des Klartextes abhängt. Bisher war es aber nicht notwendig uns sogar ausgesprochen unerwünscht, die Operation auch wieder umzukehren. Wir suchen daher nach einer Kombination von Operationen, welche umkehrbar ist, und damit als Verbzw. Entschlüsselungsverfahren eingesetzt werden kann.

Feistel-Netzwerke besitzen die Eigenschaft, dass die Umkehrbarkeit bereits durch die Konstruktion garantiert ist. Der alterwürdige DES Algorithmus, der in diesem Abschnitt beschrieben wird, verwendet ein Feistel Netzwerk.

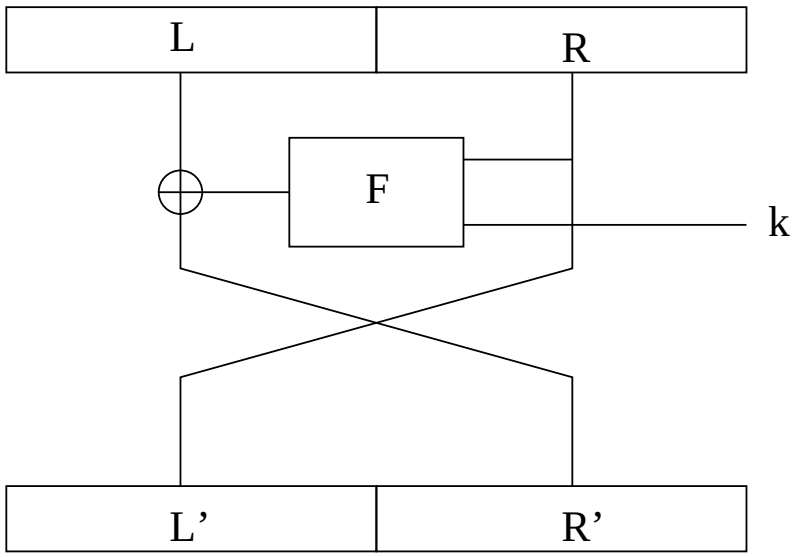

Abbildung 4.2. Feistel-Netzwerk.

4.2.1. Feistel-Netzwerke. Ein Feistel-Netzwerk beschreibt die folgende Operation auf einem Klartext-Block:

1. Der Block wird in zwei Hälften L und R aufgeteilt.
2. Der neue rechte Block R' ist $L \oplus F(R, k)$. wobei F eine vom rechten Halbblock R und einem Schlüssel abhängige Funktion ist.
3. Der neue rechte Block L' ist R ersetzt.

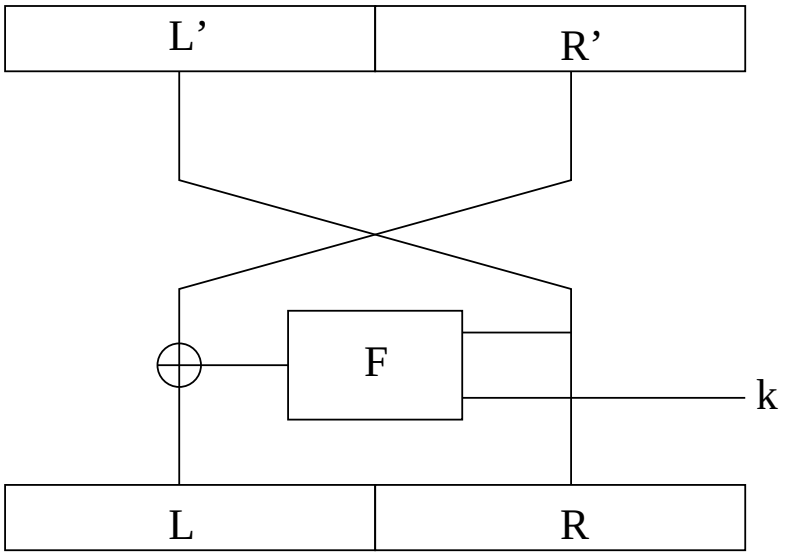

Abbildung 4.3. Invertiertes Feistel-Netzwerk.

Ein Feistel-Netzwerk kann sofort invertiert werden. Im linken Subblock steht ja der ursprüngliche rechte Subblock, welcher in die Funktion F einging. Somit ist einem Kenner des Schlüssels k auch $F(R,k) = F(L',k)$ und damit $L = R' \oplus F(L',k)$ bekannt. Die Entschlüsselung des Feistel-Netzwerkes erfolgt also mit Hilfe eines Feistel-Netzwerkes, in dem die Vertauschung dem XOR mit dem Output von F vorangeht, wie in Abbildung 4.3. gezeigt.

Bei einer Hardware-Implementation eines Feistelnetzwerkes fällt die Vertauschung der beiden Subblocks nicht ins Gewicht, da dies nur eine Verdrahtungsangelegenheit ist, es sind darin keine Gatter mit Signallaufzeiten involviert.

4.2.2. Beschreibung von DES. DES arbeitet auf Blocks von 64bit Länge. Nach einer initialen Permutation IP werden 16 Feistel-Netzwerke ausgeführt. Abgeschlossen wird der Prozess wieder mit der Inversen der initialen Permutation IP^{-1}.

Die Initiale Permutation ordnet einzelne Bits neuen Positionen zu. In Hardware ist dies trivial zu realisieren (Verdrahtung), in Software ist es jedoch ausgesprochen mühsam.

Die F-Funktion in den Feistel-Netzwerken verknüpft einen 32bit langen Subblock mit 48 Schlüssel-Bits, welche aus dem 56bit des DES-Schlüssels generiert werden. Wie bereits erwähnt rührt die Stärke eines aus Feistel-Netzwerken aufgebauten Verfahrens von den kryptographischen Qualitäten der Funktion F her.

Im Falle von DES müssen wir die Erzeugung von Teilschlüsseln aus dem DES-Schlüssel und die Verknüpfung mit dem Subblock beide als Bestandteile der F-Funktion ansehen. Der 32bit breite Subblock wird zunächst durch die Expansionstransformation auf 48bit ausgeweitet. Die S-Boxen bringen als nichtlineare Transformation Diffusion in das Verfahren, in diesem Schritt werden aus den 48bit wieder 32bit breite Subblocks. Die abschliessende P-Box sorgt dafür, dass Bitänderung schnell über grosse Distanzen innerhalb des Subblockes propagiert werden. Die folgenden Abschnitte besprechen die einzelnen Teile der F-Funktion.

Schlüsselgenerierung. Ein DES-Schlüssel besteht aus 64 bit, davon wird aber jedes achte (Paritäts-)bit ignoriert, so dass die effektive Schlüssellänge nur noch 56 ist. In jeder Runde werden aus den 56 bit Rundenschlüssel von 48 bit generiert. Dazu wird der Schlüsselblock in zwei Sub-

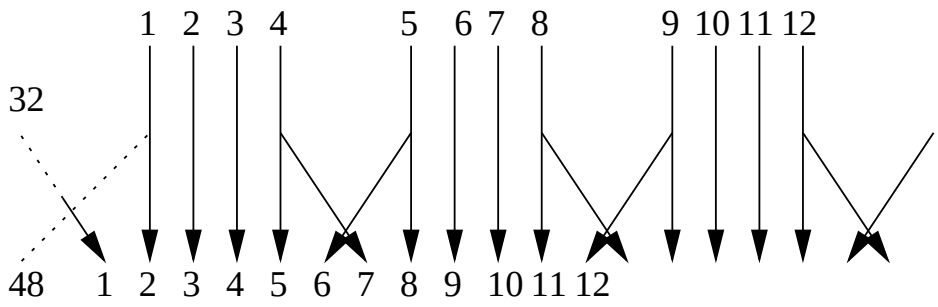

Abbildung 4.4. Expansionstransformation.

blöcke von 28 bit Länge aufgeteilt. Jeder Subblock wird abhängig von der Runde[4] zyklisch nach links rotiert. Der Rundenschlüssel besteht jetzt aus der immer gleichen, permutierten Auswahl von Bits. In Hardware ist dieser Schritt, die sogenannte Kompressions-Permutation sehr effizient zu implementieren, da er wieder nur eine Verdrahtungsangelegenheit ist.

Expansion. Die Expansionstransformation macht aus Blöcken von 4 bit solche mit 6 bit. Gleichzeitig werden jeweils die äussersten beiden Bits benachbarter 6bit-Blocks ausgetauscht. Graphisch lässt sich das wie in Abbildung 4.4. darstellen. Durch die Expansion wird sichergestellt, dass Einzelbitänderungen sich in mehreren 6bit-Blöcken des expandierten Subblockes niederschlagen können. Dies geschieht allerdings nur in der Hälfte aller Fälle, so dass für weitere Diffusion das Feistel-Netzwerk genügend häufig iteriert werden muss.

***S*-Box.** Die *S*-Box ist die einzige nichtlineare Transformation. Sie verwendet die von der Expansionstransformation erzeugten 6bit-Blocks als Input, und liefert einen 4bit-Wert zurück. Die Abbildung wird durch Tabellen gegeben, nicht durch analytische Formeln.

Die Herkunft der *S*-Boxen ist nicht bekannt. Der ursprüngliche Vorschlag von IBM wurde von der National Security Agency modifiziert. Je nach der Paranoia des Betrachters hatte dies entweder den Zweck, eine Hintertür für die NSA einzubauen, oder sicherzustellen, dass IBM keine Hintertür für sich selber eingebaut hat.

Der nichtlineare Charakter der *S*-Box Transformation hat zur Folge, dass eine Einzelbitänderung sich im Mittel auf 50% der bits des gleichen

[4] In der ersten, zeiten neunten und sechzenten (letzten) Runde wird um ein bit rotiert, in allen anderen um zwei.

4bit-Blocks des Subblocks auswirkt. Wegen der Expansionstransformation wirkt er sich in der Hälfte aller Fälle auch noch auf ebensoviele Bits in einem benachbarten Block aus. Man kann daher davon ausgehen, dass sich nach 16 Runden eine Einzelbitänderung wie gewünscht auf alle Blocks eines Subblocks ausgewirkt hat.

Für die genauen Werte der S-Boxen konsultiere man ein Standardwerk wie zum Beispiel [4].

***P*-Box.** Wie die Initiale Permutation ist auch diese Permutation eine Operation auf Bit-Ebene, ist also in Hardware sehr schnell, in Software mühsam zu realisieren. Die Permutation bringt die Bits an die folgenden Positionen:

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

4.2.3. Performance von DES. Der bereits einmal erwähnte 266MHz Pentium MMX schafft[5] knapp 6MB pro Sekunde, könnte also ein 10Mbps Ethernet noch problemlos verschlüsseln. Ein aktueller Prozessor kann also auch ein Fast-Ethernet verschlüsseln. Dass in der Praxis oft wesentlich geringere Durchsätze erreicht werden, hat mit dem Overhead für die Protokolle und mit nicht optimalen Implementationen zu tun. Eine etwas angejahrte SPARCstation 20 mit 50MHz Prozessor muss sich mit 326kB/s begnügen.

4.3. AES.

Der Rijndael-Algorithmus wurde als neuer Verschlüsselungsstandard AES (Advanced Encryption Standard) gewählt. Während einige seiner direkten Konkurrenten von einer Feistel-Struktur ausgehen, verwendet AES einen neueartigen Ansatz, bei dem algebraische Operationen in verschiedenen algebraischen Gruppen (die wegen der Existenz von Inversen Elementen automatisch umkehrbar sind) gemischt werden. Damit werden Ideen aufgenommen, die auch schon der IDEA-Algorithmus aufbrauchte, aber konsequenter und vor allem auf modernen Prozessoren effizienter implementiert.

In den folgenden Abschnitten beschreiben wir die einzelnen algebraischen Operationen, die im letzen Unterabschnitt zu einer Beschreibung der Rundenoperation des AES zusammengeführt werden.

[5] Die hier angegebenen Werte wurden mit der `speed` Funktion von Openssl ermitteln.

4.3.1. Multiplikation in $\mathbb{F}_{2^8}$. Im ersten Schritt der Rundentransformation von AES wird auf einzelnen Bytes operiert. Die Bytes werden als Polynome über $\mathbb{F}_2$ vom Grad 7 betrachtet. Eine vernünftige algebraische Struktur erhalten wir, indem wir in $\mathbb{F}_2[X]$ Reste bei Teilung durch das Polynom

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

betrachten, also den Ring $\mathbb{F}_2[x]/m\mathbb{F}_2[x]$. Da m irreduzibel (ein Primelement) ist, ist der Restklassenring ein Körper, jedes von 0 verschiedene Element in $\mathbb{F}_{2^8} = \mathbb{F}_2[x]/m\mathbb{F}_2[x]$ ist invertierbar.

Die Byte-Transformation des AES bildet ein von Null verschiedenes Byte auf das multiplikative Inverse in $\mathbb{F}_{2^8}$ ab, 0 wird auf sich selbst abgebildet. Damit ist sichergestellt, dass eine Einzelbitänderung sich durch die Bytetransformation auf alle bits des Byte, in dem sie aufgetreten ist, auswirken kann.

Wir schreiben Elemente aus $\mathbb{F}_{2^8}$ als hexadezimale Bytes in der Form

$$\texttt{0x21} = x^5 + 1.$$

Man beachte, dass die Addition und die Subtraktion (die mit der Addition identisch ist) durch einfaches und schnelles XOR realisiert werden können.

4.3.2. Multiplikation in $\mathbb{F}_{2^8}[X]/(X^4+1)\mathbb{F}_{2^8}$. Um die Diffusion zu erhöhen, betrachtet AES jeweils 4 Bytes, die wie vorher beschrieben als Zahlen in $\mathbb{F}_{2^8}$ aufgefasst werden, als Koeffizienten eines Polynomes vom Grad 3 mit Koeffizienten in $\mathbb{F}_{2^8}$.

Wiederum kann durch geeignete Restklassenbildung bezüglich eines Polynoms $M(X) \in \mathbb{F}_{2^8}[X]$ eine interessante algebraische Struktur erzeugt werden. In diesem Schritt ist es jedoch nicht notwendig, soweit zu gehen, dass alle nicht verschwindenden Polynome invertierbar werden. Es genügt, dass die Multiplikation mit einem festen Polynom invertierbar wird. Wie früher bei der Diskussion des erweiterten euklidischen Algorithmus erläutert, genügt dazu, dass der Multiplikator teilerfremd zum Polynom M ist.

Das bei AES verwendete Polynom $M(X) = \texttt{0x01}X^4 + \texttt{0x01}$ ist natürlich nicht irreduzibel, doch es ist relativ prim zu

$$c(X) = \texttt{0x03}X^3 + \texttt{0x01}X^2 + \texttt{0x01}X + \texttt{0x02}.$$

Die Multiplikation mit c in dem Ring $\mathbb{F}_{2^8}[X]/M\mathbb{F}_{2^8}[X]$ ist invertierbar, die Umkehrabbildung entspricht der Multiplikation mit

$$c(X)^{-1} = d(X) = \texttt{0x0b}X^3 + \texttt{0x0d}X^2 + \texttt{0x09}X + \texttt{0x0e}.$$

Diese Operationen sind durch Matrixmultiplikationen der Koeffizientenvektoren effizient durchführbar.

Diese Multiplikation stellt wegen der Tatsache, dass alle Koeffizienten des Polynoms $c(X)$ nicht verschwinden, sicher, dass sich Einzelbitänderungen auf das ganze 32bit-Wort verteilen.

4.3.3. Beschreibung von AES. Die Details zur Konstruktion von AES/Rijndael findet man im Rijndael-Paper [3]. AES kann Blöcke zwischen 128 und 256 bit Länge verarbeiten, und verwendet eine zweidimensionale Darstellungs in Form einer $4 \times n$-Matrix dafür. Die Daten des zu verarbeitenden Blockes werden dabei in der linken oberen Ecke beginnend nacheinander vertikal in die Kolonnen eingefüllt. Auf diesem Zustand erfolgen anschliessend die Rundenoperationen.

AES verwendet eine Anzahl von Runden, die von der Anzahl der Bytes im Datenblock und im Schlüssel abhängt, sie variiert zwischen 10 und 14 Runden. Jeder Runde besteht aus den folgenden Schritten

Bytesubstitution. Jedes nicht verschwindende Byte wird durch sein multiplikatives Inverses in $\mathbb{F}_{2^8}$ ersetzt, 0 geht in 0 über.

Zeilenverschiebung. Die Zeilen der Zustandsmatrix werden horizontal rotiert. Die Anzahl der Rotationsschritte hängt von der Zeile und von den Dimensionen der Zustandsmatrix ab.

Kolonnendurchmischung. Jede Kolonne wird als Polynom in

$$\mathbb{F}_{2^8}[X]/M\mathbb{F}_{2^8}[X]$$

aufgefasst und in diesem Ring mit dem invertierbaren Polynom $c(X)$ multipliziert.

Rundenschlüssel addieren. Der aus dem Schlüsselmaterial gewonnen Rundenschlüssel wird mit XOR zur Zustandsmatrix hinzuadiert.

In der letzten Runde wir die Kolonnenmischung weggelassen. Man kann sehr schön erkennen, wie diese Abfolge von Operationen die Diffusion verstärkt und die nichtlinearen Transformationen sehr schnell auf eine grosse Zahl von Bytes Einfluss nehmen.

4.4. Feedback-Modes.

Die Anwendung eines Verschlüsselungsverfahrens wie DES oder AES auf einzelne Klartext-Blocks ohne Einbezug des Kontext wird der Elektronische Codebuch-Mode ECB genannt. Gleich Klartextblocks führen zum gleichen Chiffrat-Block. Diese Schwäche kann vermieden werden, wenn das Chiffrat oder der Klartext eines Blockes in die Verschlüsselung des Nachfolgeblockes einfliesst. Damit wir die Diffusion mindestens auf den Nachbarblock ausgeweitet.

Die folgenden Betriebsarten oder Modi von Verschlüsselungsverfahren werden unterschieden:

Electronic Codebook Mode – ECB: Jeder Klartextblock wird unabhängig von allen anderen mit E_k verschlüsselt:

$$C_i = E_k(P_i).$$

Cipher Block Chaining Mode – CBC: Die Verschlüsselung wird auf die XOR-Summe von Klartextblock und vorangehendem Chiffratblock angewendet. Bei der ersten Verschlüsselung wird anstelle des eingespeisten Chiffratblockes ein dem Empfänger ebenfalls bekannter sogenannter Initialisierungsvektor verwendet.

$$C_i = E_k(P_i \oplus C_{i-1}).$$

Cipher Feedback Mode – CFB: Der Klartextblock wird verschlüsselt, indem das Chiffrat unter E_k des vorangehenden Chiffratblockes mit XOR hinzuaddiert wird.

$$C_i = P_i \oplus E_k(C_{i-1}).$$

Output Feedback Mode – OFB: Aus einem Initialisierungsvektor wird durch wiederholte Anwendung von E_k eine Folge von "Zufallsblöcken" S_i generiert. Der i-te Klartextblock P_i wird verschlüsselt, indem S_i mit XOR hinzuaddiert wird:

$$S_i = E_k(S_{i-1}), \qquad C_i = P_i \oplus S_i.$$

4.5. Stromverschlüssler.

Ausser den bisher besprochenen Verfahren, die jeweils auf einem Block fester Länge operieren, gibt es auch Verfahren, die Byte um Byte in einem Datenstrom verschlüsseln können. Da auf dem Niveau dieses Scripts keine neuen Erkenntnisse aus der Besprechung der Stromverschlüssler zu gewinnen sind, wird der interessierte Leser auf die Literatur [4] verwiesen.

4.6. Anwendungen.

Auch mit symmetrischen Verschlüsselungsverfahren lassen sich interessante Anwendungen bauen. Als Beispiel betrachten wir das Authentisierungsverfahren von Needham und Schroeder.

4.6.1. Needham-Schroeder. Mit diesem Verfahren kann sich A bei B mit Hilfe eines vertrauenswürdigen Dritten T authentisieren. Jeder Teilnehmer hat mit T einen geheimen Schlüssel, Entschlüsselung mit und Verschlüsselung für den Teilnehmer P werden mit E_P und D_P bezeichnet.

1. A wählte eine Zufallszahl r_A und übermittelt

   $$A, B, r_A$$

   an T.
2. T generiert einen zufälligen Sitzungsschlüssel K und übermittelt

   $$E_A(r_A, B, K, E_B(K, A))$$

   an A
3. A Entschlüsselt die Meldung. Sofern A den Schlüssel (das Passwort) nicht kennt, fällt A in diesem Moment auf die Nase. Anschliessend sendet A den Teil

   $$E_B(K, A)$$

   an B.
4. B Entschlüsselt die Meldung und findet darin K. Jetzt wählt B eine Zufallszahl r_B und übermittelt

   $$E_K(r_B)$$

an A.

5. A übermittelt
$$E_K(r_B - 1)$$
an B.

Im letzten Schritt wird von r_B 1 subtrahiert um dem Verschlüssler E_K garantiert einen anderen Input zu geben, man schützt sich so vor Replay-Attacken auf diesen Schritt.

Falls ein Angreifer K stehlen kann, kann er B voräuschen, A zu sein. Er steigt dazu einfach im Schritt 3 ein und übermittelt die Message $E_B(K, A)$, die bereits einmal auf dem Netzwerk sichtbar gewesen ist. B wird mit einer mit K verschlüsselten Meldung antworten, die dem Angreifer auch ohne Kenntnis der Schlüssel E_A und E_B zugänglich sind. Man könnte sich gegen diesen Angriff schützen, indem die Meldungen an B mit einem sicheren Timestamp versehen werden, oder indem B über die erhaltenen $E_B(K, A)$ Buch führt und Duplikate verwirft.

Kapitel 5. Asymmetrische Verfahren

In diesem Kapitel werden die asymmetrischen Verschlüsselungsverfahren beschrieben, die den verbreiteten Internet-Protokollen zu Grunde liegen. Diese Verfahren sind zwar sehr elegant, ihr praktischer Nutzen wird dagegen durch den relativ grossen Rechenaufwand und die vergleichsweise langen Schlüssel eingeschränkt. Asymmetrische Verfahren werden daher praktisch ausschliesslich in Kombination mit symmetrischen Verfahren verwendent: zunächst wird mit Hilfe eines asymmetrischen Verfahrens ein Sitzungsschlüssel etabliert und gegebenenfalls die Kommunikationsteilnehmer authentisiert, die anschliessende Kommunikation verwendet jedoch ausschliesslich das symmetrische Verfahren.

Alle in diesem Kapitel vorgestellten Verfahren basieren auf den mathematischen Eigenschaften grosser Primzahlen, die wichtigsten Sätze werden im ersten Abschnitt zusammengestellt. Im zweiten Abschnitt wird die Arithmetik mit grossen Zahlen vertieft, die einen wesentlichen Beitrag zu einer effizienten Implementation leistet. Die folgenden Abschnitte behandeln das Diffie-Hellman Verfahren und einige Verwandte, sowie das nach den Erfindern Ron Rivest, Adi Shamir und Leo Adleman benannte RSA Verfahren. Für letzteres müssen die mathematischen Aussagen nochmals erweitert werden. Der letzte Abschnitt behandelt die durch wesentlich verbesserte Performance sich auszeichnenen, im wesentlichen aber zu Diffie-Hellman analogen Verfahren in elliptischen Kurven.

5.1. Zahlentheoretische Basis.

Die verbreiteten asymmetrischen Verfahren basieren auf der Schwierigkeit der folgenden beiden zahlentheoretischen Probleme:

Diskreter Logarithmus: Ist p eine grosse Primzahl, so ist es im Allgemeinen schwierig, aus $g^x \mod p$ für ein $g \in \mathbb{F}_p$ die Zahl x zu ermitteln.

Faktorisierung eines Produktes grosser Primzahlen: Sind p_1 und p_2 grosse Primzahlen, dann ist es im Allgemeinen schwierig, aus dem Produkt $n = p_1 p_2$ die Faktoren zu ermitteln.

Für beide Probleme gibt es keinen mathematischen Beweis, etwa im Sinne der Komplexitätstheori, dass sie tatsächlich schwierig sind. Nach heutigem Wissensstand gibt es keine praktikablen Methoden, die genann-

ten Probleme innert nützlicher Frist zu lösen, sofern nur die beteiligten Primzahlen gross genug sind. Doch ist nicht völlig auszuschliessen, dass eine fundamentale neue Entdeckung der Zahlentheorie nicht das eine oder andere Verfahren zu Fall bringen könnte.

5.1.1. Grosse Primzahlen. Für Anwendungen sowohl des diskreten Logarithmus wie auch der Faktorisierung werden grosse Primzahlen benötigt, gemeint sind dabei Primzahlen mit mindestens 1000 bit. Nun lassen sich die kleinen Primzahlen sehr gut aufzählen, es ist jedoch nicht mehr so einfach, grosse Primzahlen zu finden. Zum Einen ist nicht klar, wo man überhaupt danach suchen soll, wegen der vielen möglichen Teiler werden grosse Primzahlen im Mittel weiter auseinander liegen als kleine. Zum anderen müssen wir bei einem gefundenen Kandidaten zweifelsfrei feststellen können, ob er wirklich eine Primzahl ist. Die Methode der Faktorisierung scheidet dabei offensichtlich aus, genau die praktische Unmöglichkeit dieses Problems wollen wir ja ausnutzen.

Verteilung von Primzahlen. Die Zahl der Primzahlen, die kleiner als x sind, heisst $\pi(x)$. Über diese Funktion gibt es eine Reihe von Resultaten, für uns genügt jedoch der Gausssche Primzahlsatz:

Satz 5.1 (Gauss). *$\pi(x)$ verhält sich asymptotisch wie $x/\log x$.*

Dieser Satz liefert leider nur eine statistische Aussage, so kann man zum Beispiel ausrechnen, dass zwischen 10^6 und $2\cdot 10^6$ wegen

$$\begin{aligned}\frac{2\cdot 10^6}{\log 2+6\log 10}-\frac{10^6}{6\log 10}&=\left(\frac{2}{\log 2+6\log 10}-\frac{1}{6\log 10}\right)\cdot 10^6\\&\simeq 65466.3133677738\end{aligned}$$

etwa 65000 Primzahlen zu finden sein könnten, doch gibt es keine Garantie, dass überhaupt eine Primzahl vorhanden ist. Ähnlich sieht es zwischen 10^{300} und $2\cdot 10^{300}$ aus:

$$\left(\frac{2}{\log 2+300\log 10}-\frac{1}{300\log 10}\right)\cdot 10^{300}=1.44474594827\cdot 10^{297},$$

mehr als jede Tausendste Zahl ist ein Primzahl. Die Gewissheit, dass eine Primzahl gefunden werden kann, liefert der folgende Satz von Bertrand:

Satz 5.2 (Bertrand). *Zwischen $n \geq 2$ und $2n$ gibt es immer mindestens eine Primzahl p mit $n \leq p < 2n$.*

Damit liesse sich nun ein praktikabler Algorithmus für die Bestimmung einer Primzahl einer bestimmten Grösse angeben: man beginnt bei einer zufällig gewählten Zahl n, und testet alle Zahlen n, $n+1$, $n+2$, ..., bis man eine Primzahl gefunden hat. Sofern die Ausgangsgrössen zufällig waren, ist unwahrscheinlich, dass zwei Personen auf die gleiche Primzahl stossen. Bei tausendstelligen Primzahlen muss man im Schnitt etwa 350 Zahlen durchtesten, bis man auf eine Primzahl trifft. Doch wie testet man, ob eine Zahl prim ist?

Primzahltests. Die Faktorisierung ist für grosse Primfaktoren kein praktikables Verfahren, um Faktoren zu ermitteln.

Wir sind aber gar nicht an der Bestimmung der Faktoren interessiert, es genügt zu wissen, ob eine Zahl prim ist oder nicht. Ja es ist nicht einmal notwendig, eine Aussage mit absoluter Gewissheit zu haben. Es genügt, wenn die Wahrscheinlichtkeit sehr gering ist, dass eine als prim vermutete Zahl trotzdem Faktoren hat.

Satz 5.3 (Lehmann). *Der folgende Algorithmus erlaubt, die Wahrscheinlichkeit abzuschätzen, dass die Zahl p nicht prim ist:*

1. *Wähle zufällig eine Zahl $a < p$.*
2. *Berechne $x = a^{(p-1)/2} \in \mathbb{F}_p$.*
3. *Falls $x \neq -1$ und $x \neq 1$ in $\mathbb{F}_p$, dann ist p definitiv keine Primzahl.*
4. *Falls $x = 1$ oder $x = 1$ in $\mathbb{F}_p$, dann ist die Wahrscheinlichkeit, dass p keine Primzahl ist, kleiner als 50%.*

Wird der Algorithmus mit t verschiedenen Werten für a wiederholt, wobei jedes mal der Fall 4 eintrifft, dann ist die Wahrscheinlichkeit, dass p prim ist, kleiner als 2^{-t}.

Beweis: Der Beweis setzt die Theorie der Legendre-Symbole voraus, die zu entwickeln uns die Zeit fehlt.

Ein noch effizienterer Test ist

Satz 5.4 (Rabin-Miller). *Die Zahl p ist daraufhin zu testen, ob Sie prim ist. Sei b der grösste Exponent so, dass $2^b | (p-1)$, und $m = (p-1)2^{-b}$. Dann führe folgenden Algorithmus aus:*

1. *Wähle eine beliebige Zahl* $0 \leq a < p$.
2. *Setze* $j = 0$ *und* $z = a^m \mod p$.
3. *Falls* $z = 1$ *oder* $z = p - 1$, *dann besteht* p *den Test, und könnte prim sein.*
4. *Falls* $j > 0$ *und* $z = 1$, *dann ist* p *keine Primzahl.*
5. *Setze* $j = j + 1$. *Falls* $j < b$ *und* $z \neq p - 1$, *setze* $z = z^2 \mod p$ *und wiederhole Schritt (4). Falls* $z = p - 1$, *besteht* p *den Test und könnte prim sein.*
6. *Falls* $j = b$ *und* $z \neq p - 1$, *dann ist* p *keine Primzahl.*

Falls p den Test besteht ist die Wahrscheinlichkeit, dass p keine Primzahl ist, kleiner als $\frac{1}{4}$. Nach t Iterationen des Tests ist die Wahrscheinlicht also noch 2^{-2t}.

5.1.2. Der klein Satz von Fermat. Die Menge der Potenzen $\langle x \rangle = \{x^n | n \in \mathbb{N}\} \subset \mathbb{F}_p^*$ ist eine Teilmenge, die bezüglich der Multiplikation abgeschlossen ist. Man kann normalerweise nicht erwarten, dass $\langle x \rangle$ die ganze Gruppe $\mathbb{F}_p^*$ ist. Jedoch gilt immer:

Satz 5.5 (Fermat). *Falls* p *ein Primzahl ist, gilt* $a^p = a$ *in* $\mathbb{F}_p$.

Beweis: Die Aussage des Satzes ist für $a = 0$ klar. Betrachten wir jetzt also $a \neq 0$. Dann sind die Zahlen

$$0, a, 2a, 3a, \ldots, (p-1)a \in \mathbb{F}_p$$

alle verschieden, denn wären zwei gleich, zum Beispiel $x_1 a$ und $x_2 a$, dann wäre $(x_1 - x_2)a = (x_2 - x_1)p = 0$. Da aber p teilerfremd zu a ist, müsste $p|(x_1 - x_2)$ sein, was nur geht, wenn $x_1 = x_2$. Da aber $\mathbb{F}_p$ genau p Element hat, muss die Liste $0, a, 2a, \ldots, (p-1)a$ alle Elemente von $\mathbb{F}_p$ umfassen. Die Elemente $a, 2a, \ldots, (p-1)a$ sind also genau die Zahlen $1, 2, \ldots, (p-1)$, nur in anderer Reihenfolge. Das Produkt dieser Zahlen ist also

$$a^{p-1}(p-1)! = (p-1)!$$

Da aber p und $(p-1)!$ teilerfremd sind, folgt auch

$$a^{p-1} = 1 \in \mathbb{F}_p,$$

oder

$$a^p = a.$$

Damit ist der kleine Satz von Fermat bewiesen.

5.2. Arithmetik mit grossen Zahlen.

In den Anwendungen muss mit grossen Ganzzahlen von einigen Tausend bit gerechnet werden können. Spätestens beim Potenzieren wird offensichtlich, dass die "Schulverfahren" nicht durchführbar sind. So würden bei der Berechnung von g^x nach dem klassischen Verfahren x Multiplikationen notwendig, ein Aufwand, der exponentiell mit der Länge von x in bits anwächst. Realistisch sind dagegen nur Verfahren, für die der Aufwand schlimmstenfalls mit einer kleinen Potenz der Länge von x ansteigt.

Praktisch werden grosse ganze Zahlen als Folgen (Arrays) von Zahlen von der Grösse eines primitiven Datentyps der Maschine realisiert. Bei einer Wortlänge von l bit hat man sich die einzelnen Elemente des Array als "Ziffern" einer 2^l-adischen Darstellung vorzustellen. Eine Zahl x ist mit Arrays der Länge $\log_2 x/l$ darstellbar. Für die folgende Diskussion gehen wir von Zahlen aus, die mit einem Array der Länge L darstellbar sind.

5.2.1. Reduktion. In den Anwendungen interessiert die Arithmetik in einem Ring $\mathbb{Z}/n\mathbb{Z}$. Der klassische Divisionsalgorithmus würde in der Basis 10 die Multiplikation voraussetzen. In der Basis 2 hingegen können als Teilquotienten nur 0 oder 1 auftreten, je nach dem, ob der aktuelle Rest grösser oder kleiner ist als der Divisor. Die notwendigen Multiplikationen sind also trivial. Der Aufwand für die Reduktion ist daher von der Grössenordnung von $l \cdot L$ Additionen.

5.2.2. Addition und Subtraktion. Das übliche Verfahren der Addition oder Subtraktion mehrstelliger Zahlen erfordert linearen Aufwand in $\log_2 x$, es sind keine Verbesserungen erkennbar. Typischerweise sind $2L - 1$ Additionen erforderlich (je eine Addition für die Elemente der Vektoren, sowie eine zusätzliche Addition für den Übertrag, die beim niederwertigsten Element nicht nötig ist.

5.2.3. Multiplikation. Die Multiplikation erfordert üblicherweise L^2 Multiplikationen und ebensoviele Additionen. Der folgende Trick von Karatsuba erlaubt jedoch, die Zahl der Multiplikationen zu reduzieren:

Satz 5.6 (Karatsuba). *Die Multiplikation von zwei Zahlen mit $2n$ Bits lässt sich mit nur drei Multiplikationen von Zahlen mit n Bits realisieren.*

Beweis: Wir schreiben die Zahlen u und v als

$$u = 2^n U_1 + U_0, \qquad v = 2^n V_1 + V_0,$$

wir bilden also eine s^n-adische Darstellung von u und v. Für das Produkt uv gilt jetzt

$$\begin{aligned} uv &= 2^{2n} U_1 V_1 + 2^n (U_1 V_0 + V_1 U_0) + U_0 V_0 \\ &= 2^{2n} U_1 V_1 + 2^n (U_1 - U_0)(V_0 - V_1) + 2^n U_1 V_1 + 2^n U_0 V_0 + U_0 V_0 \\ &= (2^n + 1) 2^n U_1 V_1 + 2^n (U_1 - U_0)(V_0 - V_1) + (2^n + 1) U_0 V_0 \end{aligned}$$

Wichtig dabei ist, dass die Multiplikation von x mit 2^n+1 durch Addition eines um n Bit nach links verschobenen x zu x realisiert werden kann, also ohne Multiplikation.

Für Zahlen der Länge 2^{2^l} Worte lässt sich die Multiplikation also mit 3^l Wortmultiplikationen durchführen, oder mit $3^{\log_2 \log_2 x} = (\log_2 x)^{\log_2 3} \simeq (\log_2 x)^{1.5849}$ statt $(\log_2 x)^2$ Multiplikationen. Der Nutzen zeigt sich vor allem bei sehr grossen Produkten. So ist zum Beispiel bei einem Produkt von 2 Zahlen von je 1024 Bit der Karatsuba-Prozess 5 mal anwendbar, somit ist die Multiplikation mit $3^5 = 243$ Wortmultiplikationen möglich statt mit $32^2 = 1024$, eine Reduktion auf einen Viertel!

5.2.4. Division in $\mathbb{Z}/n\mathbb{Z}$. In einem Ring $\mathbb{Z}/n\mathbb{Z}$ ist das Element x genau dann invertierbar, wenn x und n teilerfremd sind. Das multiplikative Inverse von x findet man mit Hilfe des erweiterten euklidischen Algorithmus. Dieser benötigt in jedem Schritt eine Division mit Rest (Reduktion), zwei Multiplikationen und zwei Additionen. Der Aufwand wird daher dominiert von den Multiplikationen.

5.2.5. Potenzieren. Sei der Exponente x im Binärsystem als

$$\sum_{i=0}^{l} x_i 2^i$$

geschrieben. Um g^x zu berechnen, verwendet man

$$g^{\sum_{i=0}^{l} x_i 2^i} = \prod_{i=0}^{l} g^{x_i 2^i} = \prod_{i=0, x_i=1}^{l} g^{2^i}.$$

Die Potenzen g^{2^i} sind dabei mit nur l Multiplikationen zu berechnen, da $g^{2^i} = (g^{2^{i-1}})^2$ ist. Die totale Anzahl von Multiplikationen ist daher $l + \sum_{i=0}^{l} x_i \leq 2l$, also linear in der Länge des Input.

Das Element g liegt in den Anwendungen oft in einem Körper $\mathbb{F}_p$ (Diffie-Hellman) oder einem Ring $\mathbb{Z}/n\mathbb{Z}$ (RSA). Selbst wenn sich mit obigem Verfahren des Quadrierens und Multiplizierens viele Multiplikationen sparen lassen, sind trotzdem die Produkte von sehr grossen Zahlen zu bestimmen, und in jedem Schritt zu reduzieren. Die Multiplikation ist zwar dank des Tricks von Karatsuba effizient durchführbar, für Reduktion kommt man aber nicht um eine Division herum, was sehr zeitaufwendig ist. Ein Verfahren, welches die Anzahl der Reduktionen verringern könnte ist daher sehr willkommen.

5.2.6. Montgomery-Multiplikation. Peter Montgomery [2] hat festgestellt, dass sich die Reduktionen bei Rechnungen in $\mathbb{Z}/N\mathbb{Z}$ vereinfachen lassen, insbesondere bei wiederholten Multiplikationen wie etwa beim Potenzieren. Die Idee dahinter ist, dass die Reduktion sehr einfach wäre, wenn N eine Zweierpotenz wäre. Man wählt daher eine Zahl R als Zweierpotenz grösser als N und teilerfremd zu N. Da R und N teilerfremd sind, gibt es ganze Zahlen R' und N' mit

$$RR' - NN' = 1,$$

R' ist also auch eine Inverse von R in $\mathbb{Z}/N\mathbb{Z}$, also $RR' = 1 \in \mathbb{Z}/N\mathbb{Z}$. Ausserdem ist $N - N'$ die Inverse von N in $\mathbb{Z}/R\mathbb{Z}$.

Die Abbildung

$$M : \mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/N\mathbb{Z} : x \mapsto Rx$$

ist invertierbar und hat als Inverse

$$M^{-1} : \mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/N\mathbb{Z} : x \mapsto R'x.$$

$M(x)$ heisst die Montgomery-Darstellung von x. M^{-1} lässt sich mit dem folgenden Algorithmus effizient berechnen. Wir schreiben für die Berechnung des Restes jeweils %.

Algorithmus 5.7 (Mongomery's REDC-Algorithmus). *Sei* $0 \leq x < N$. *Berechne*

1. $m = ((x \,\%\, R)N') \,\%\, R$
2. $t = (x + mN)/R$
3. *Falls* $t < N$, *ist* $M'(x) = t$, *andernfalls* $M'(x) = t - N$.

Beweis: Zunächst müssen wir nachrechnen, dass die Division im zweiten Schritt überhaupt möglich ist. Aber

$$mN \,\%\, R = (((x \,\%\, R)N') \,\%\, R)N \,\%\, R = xNN' \,\%\, R = -x \,\%\, R$$
$$(x + mN) \,\%\, R = x + (-x) \,\%\, R = 0,$$

die Division geht also auf.

In $\mathbb{Z}/N\mathbb{Z}$ gilt

$$\begin{aligned} & x + mN = x \\ \Rightarrow \qquad & tR = x \\ \Rightarrow \qquad & t = tRR' = xR' \end{aligned}$$

Daraus folgt, dass t den gleichen Rest bei Teilung durch N hat wie $M'(x)$.

Nun müssen wir nur noch zeigen, dass das im zweiten Schritt gefundene t zwischen 0 und $2N - 1$ liegt, so dass die Rechnung im dritten Schritt tatsächlich einen Wert zwischen 0 und $N - 1$ zurückgibt. Es ist aber $m < R$, also auch

$$x + mN < RN + RN \qquad \Rightarrow \qquad t = \frac{x + mN}{R} < 2N,$$

was zu beweisen war.

Man beachte, dass der Algorithmus REDC viel effizienter durchführbar ist als die gewöhnliche Reduktion. Da R eine Zweierpotenz ist, ist die Reduktionen im ersten Schritt einfach das Abschneiden der Stellen grösser als R. Und die Division im zweiten Schritt ist nur ein Rechtsshift.

Beim Potenzieren müssen Produkte berechnet und sofort reduziert werden. Statt $xy \,\%\, N$ zu berechnen, könnte man $M(x)M(y) = xRyR = xyR^2 = M(xy)R$ berechnen, also

$$M(xy) = M'(M(x)M(y)).$$

Statt nach jeder Multiplikation zu reduzieren, kann man also ebenso gut die Faktoren zu Beginn mit M transformieren (dazu sind zwei richtige Reduktionen erforderlich), das Produkt der transformierten berechnen und mit Hilfe des REDC-Algorithmus M' berechnen.

Für die Berechnung der Potenz $g^x \in \mathbb{Z}/N\mathbb{Z}$ heisst das:

1. Berechne $M(g)$.
2. Berchne die Quadrate von $M(g)$ als

   $$M(g^{2^i}) = M'(M(g^{2^{i-1}})^2).$$

   Dazu ist nur eine Multiplikation und eine Anwendung von M' notwendig.
3. Berechne das Produkt derjenigen $M(g^{2^i})$, für die $x_i = 1$. Verwende dabei jeweils

   $$M(xy) = M'(M(x)M(y)),$$

   in jedem Schritt ist also eine gewöhnlich Multiplikation und eine Anwendung von M' notwendig. So erhält man $M(g^x)$.
4. Berechne $g^x = M'(M(g^x))$:

Man kann also $g^x \in \mathbb{Z}/N\mathbb{Z}$ mit zwei Reduktionen, $l + \sum_{i=0}^{l} x_i$ Multiplikationen und $l + \sum_{i=0}^{l} x_i + 1$ Anwendungen von REDC berechnen. Man spart also $l + \sum_{i=0}^{l} x_i - 2$ Reduktionen auf Kosten von $l + \sum_{i=0}^{l} x_i + 1$ Anwendungen von REDC.

Beispiel. Wir führen die Rechnung für $N = 1291$ und $R = 2048$ durch, wir wollen 5^{17} in $\mathbb{Z}/1291\mathbb{Z}$ berechnen. Zunächst findet man mit dem euklidischen Algorithmus

$$220 \cdot 2048 - 1291 \cdot 349 = 1,$$

also $R' = 220$ und $N' = 349$.

Die Basis g ist in Montgomery-Darstellung

$$M(g) = M(5) = 5 \cdot 2048 \,\%\, 1291 = 1203.$$

Der Exponent $17 = 10001_2$ hat nur zwei Einsen, wir müssen also nur $M(g^{16})$ berechnen und mit $M(g)$ multiplizieren.

Wir wenden den REDC-Algorithmus auf 1203^2 an. In Binärdarstellung ist $1203^2 = 10110000101010010010 01_2$, die Reduktion bedeutet, nur die 11 Stellen ganz rechts zu behalten, man erhält $10100101001_2 = 1321$. Damit wird $m = 1321N' \,\%\, R = 229$, womit der erste Schritt abgeschlossen ist.

Im zweiten Schritt berechnen wir

$$x + mN = 1203^2 + 229 \cdot 1291 = 1742848 = 851 \cdot 1291$$

also ist $M(g^2) = 851$.

Führt man die Rechnung weiter durch, erhält man folgende Tabelle

$$\begin{aligned} M(g) &= 1203 \\ M(g^2) &= 851 \\ M(g^4) &= 619 \\ M(g^8) &= 866 \\ M(g^{16}) &= 520 \end{aligned}$$

Das Produkt des ersten und des letzten Terms wird analog bestimmt und ergibt

$$M(g^{17}) = M(gg^{16}) = 18,$$

und daraus kann man durch nochmalige Anwendung von REDC g^{17} bestimmen

$$g^{17} = M'(18) = 87 \in \mathbb{F}_{1291}.$$

5.3. Diffie-Hellman.

Das älteste auf Eigenschaften von Primzahlen beruhende asymmetrische Verfahren ist das Schlüsselaustausch-Protokoll von Diffie und Hellman. Es hat Eingang in vielen Anwendungen gefunden, zum Beispiel in Secure Sun RPCs, in SSL und TLS.

5.3.1. Klassisches Diffie-Hellman Verfahren. Wenn zwei Partner A und B mit Hilfe von Diffie-Hellman einen gemeinsamen Schlüssel aushandeln wollen, einigen Sie sich zunächst auf eine grosse Primzahl p und ein geeignetes Element $g \in \mathbb{F}_p^*$. Sie streben an, g so zu wählen, dass die Menge $\{g^n | n \in \mathbb{N}\}$ möglichst die ganze Gruppe $\mathbb{F}_p^*$ ist. Diese kann lange vor dem eigentlichen Schlüsselaustausch geschehen, die Wahl von p und

g braucht nicht geheim gehalten zu werden, und es ist auch akzeptabel, wenn alle die gleichen p und g wählen.

Nun wählt A eine Zufallszahl $x \in \mathbb{F}_p$ und B eine Zufallszahl $y \in \mathbb{F}_p$. A berechnet (in $\mathbb{F}_p$) g^x, B berechnet g^y. Jeder teilt seinen Wert dem anderen Partner mit. A erhält also g^y, und berechnet daraus $(g^y)^x = g^{xy}$, B erhält g^x und berechnet $(g^x)^y = g^{xy}$. A und B verfügen jetzt also über ein gemeinsames Geheimnis: g^{xy}.

Wir behaupten, dass es für einen Aussenstehenden praktisch unmöglich ist, g^{xy} zu bestimmen. Dazu müsste der Angreifer nämlich x und y bestimmen können, das ist jedoch nur dann möglich, wenn er den diskreten Logarithmus, also den Logarithmus in $\mathbb{F}_p$ zur Basis g, effizient berechnen kann.

Das Verfahren von Diffie-Hellman ist jedoch nicht absolut sicher: kann ein Angreifer eine Zwischenposition (Man in the middle) einnehmen, kann er sowohl mit A und B einen Schlüsselaustausch vornehmen. Damit kann er die Daten von A mit dessen Schlüssel entschlüsseln, mit dem mit B ausgehandelten Schlüssel verschlüsseln, und umgekehrt. A und B erhalten so den Eindruck, verschlüsselt zu kommunizieren, obwohl der Angreifer in der Mitte alles mitlesen und eventuell sogar modifizieren kann.

Beispiel. Sei $p = 1291$ und $g = 3$, alle Rechnungen müssen in $\mathbb{F}_{1291}$ durchgeführt werden. Wir führen die Rechnung für $x = 47$ und $y = 49$ durch. A berechnet $3^{47} = 811$, B berechnet $3^{49} = 844$, danach werden die Werte ausgetauscht. A berechnet jetzt $844^{47} = 812$, B berechnet $811^{49} = 812$, die beiden Parteien haben also tatsächlich das gleiche Geheimnis 812 gefunden.

5.3.2. SRP. Das SRP-Verfahren korrigiert den eben beschriebenen Mangel des klassischen Diffie-Hellman-Verfahrens. SRP versucht, Benutzer und Server zu authentisieren. Der Server führt eine Datenbank von Tripeln, die wie folgt berechnet werden. Zunächst wird aus Username, Passwort und einem zufälligen salt die Grösse

$$x = \mathrm{SHA}(\mathrm{salt}, \mathrm{SHA}(\mathrm{Username}, \text{“:”}, \mathrm{Passwort}))$$

berechnet. Daraus werden die Tripel

$$(\mathrm{Username}, v = g^x \mod p, \mathrm{salt})$$

abgeleitet.

Zur Authentisierung bestimmt A eine Zufallszahl a, und übermittelt Benutzername und $A = g^a \mod p$ an den Server. Der Server B wählt aus der Datenbank den Verifier v, und generiert seinerseits eine Zufallszahl b. Er berechnet $B = (v + g^b) \mod p$. Das salt aus der Datenbank und B werden an A übermittelt. Jetzt kann jeder der Partner die Grösse S bestimmen, und zwar verwendet A die Formel

$$S = (B - g^x)^{(a+ux)} \mod p,$$

B verwendet

$$S = (Av^u)^b \mod p.$$

u ist eine 32bit Zahl, gebildet aus den ersten 32bit das SHA1-Hashwertes von B. An dieser Stelle haben sich Client A und Server B auf ein gemeinsames Geheimnis S geeinigt, sofern der auf Serverseite gespeicherte Verifier identisch ist mit dem auf Clientseite berechneten. Daraus leiten Sie mit Hilfe einer speziellen Hashfunktion den Schlüssel K ab. Zum Abschluss der Authentisierung müssen die Parteien also noch beweisen, dass Sie zum gleichen Schlüssel gekommen sind. Dazu berechnet der Client

$$M_C = H(H(p) \oplus H(g), H(\text{Username}), \text{salt}, A, B, K),$$

und übermittelt M_C an den Server. Dieser kann seinerseits M_S berechnen, da alle in M_C einfliessenden Parameter auch ihm bekannt sind. Stimmen M_S und M_C überein, kann der Server davon ausgehen, dass der Client auf den gleichen Schlüssel K gekommen ist, was beweist, dass er das korrekte Passwort kennt. Der Server berechnet jetzt $H(A, M, K)$, und übermittelt diese Grösse dem Client. Der Client kann diese Grösse ebenfalls berechnen, und kann schliessen, dass einerseits sein M vom Server akzeptiert wurde, und andererseits, bei Übereinstimmung, dass der Server den gleichen Schlüssel K wie der Client verwendet. Damit weiss der Client, dass der Server im Besitz des Authentisierungs-Tripels ist, also höchstwahrscheinlich echt ist.

5.3.3. Verschlüsselung mit ElGamal. Um eine Mitteilung M zu verschlüsseln, werden in $\mathbb{F}_p$ $a = g^k$ und $b = y^k M$ berechnet. Das Paar (a, b) bildet das Chiffrat, es ist doppelt so lang wie der ursprüngliche Klartext.

k ist eine beliebige Zahl teilerfremd zu $p-1$, $y = g^x$. Um die Meldung wieder zu entschlüsseln, berechnet man zunächst die Inverse von $a^x \in \mathbb{F}_p$, was mit dem euklidischen Algorithmus problemlos möglich ist. Dann gilt

$$b(a^x)^{-1} = y^k M(a^x)^{-1} = g^{kx} M(a^x)^{-1} = (g^k)^x M(a^x)^{-1} = M.$$

5.4. RSA.

Die Publikation des Verfahrens von Rivest, Shamir und Adleman hat nicht nur wegen dem mathematischen Inhalt Staub aufgewirbelt, die politischen und rechtlichen Fragen rund um die Veröffentlichung haben mindestens im gleichen Mass dazu beigetragen, das Verfahren einem breiteren Publikum bekannt zu machen. Im Gegensatz zu Diffie-Hellman ist RSA ein richtiges Verschlüsselungsverfahren, insbesondere muss ausser der Verschlüsselung auch die Entschlüsselung definiert werden.

5.4.1. Erweiterung des Satzes von Fermat. Der kleine Satz von Fermat liefert eine Identität für alle Elemente von $\mathbb{Z}/p\mathbb{Z} \setminus \{0\}$. In $\mathbb{Z}/n\mathbb{Z}$, kann man das nicht mehr erwarten.

Eulers φ-Funktion. Mit $\varphi(n)$ bezeichnen wir die Anzahl der Zahlen x mit $0 < x < n$, welche zu n teilerfremd sind.

Satz 5.8. *Genau dann wenn a und n teilerfremd sind, sind die Vielfachen*

$$0, a, 2a, 3a, \ldots, (n-1)a$$

von a alle verschieden, und decken somit die ganze Menge $\mathbb{Z}/n\mathbb{Z}$ ab.

Man sagt auch, a ist ein Erzeuger der additiven Gruppe $\mathbb{Z}/n\mathbb{Z}$. Wir werden diesen Satz später dazu verwenden, den allgemeinen kleinen Satz von Fermat zu beweisen.

Beweis: Falls a und n teilerfremd sind, sind die genannten Vielfachen von a verschieden. Gäbe es nämlich s und t mit $sa = ta$, dann müsste $(s-t)a$ durch n teilbar sein. Da n und a teilerfremd sind, müsste also auch $(s-t)$ durch n teilbar sein, was nach Voraussetzung gar nicht möglich ist. Wenn aber alle Elemente von $\{ka | 0 \leq k < n-1\} \subset \mathbb{Z}/n\mathbb{Z}$ verschieden sind, müssen die beiden Mengen übereinstimmen:

$$\{ka | 0 \leq k < n-1\} = \mathbb{Z}/n\mathbb{Z}.$$

Nehmen wir hingegen an, dass n und a einen gemeinsamen Teiler k haben, dann gibt es auch l mit $a = kl$ und $m < n$ mit $n = mk$. Dann ist auch $ma = mkl = nl$ durch n teilbar, somit enhält die Liste der Zahlen

$$0, a, 2a, 3a, \ldots, (n-1)a$$

die Null mindestens zwei mal, kann also nicht die ganze Menge $\mathbb{Z}/n\mathbb{Z}$ abdecken. Damit ist der Satz bewiesen.

Wir versuchen nun die Funktion φ zu berechnen. Zunächst ist bereits bekannt, $\varphi(p) = p-1$, denn alle Zahlen $1, \ldots, p-1$ sind zu p teilerfremd.

SATZ 5.9. *$\varphi(p^n) = (p-1)p^{n-1}$ für alle Primzahlen p und $n > 0$.*

BEWEIS: Gemäss Definition von $\varphi(p^n)$ müssen wir bestimmen, wieviele der Zahlen

$$1, \ldots, p^n - 1$$

zu p teilerfremd sind. Offensichtlich sind genau die Vielfachen von p nicht zu p teilerfremd, also die Zahlen $p, 2p, 3p, \ldots, (p^{n-1}-1)p$. Damit haben wir genau $p^{n-1} - 1$ von $p^n - 1$ Zahlen ausgeschieden, also bleiben

$$\varphi(p^n) = p^n - 1 - (p^{n-1} - 1) = p^n - p^{n-1} = (p-1)p^{n-1}$$

Zahlen, womit der Satz bewiesen ist.

Wenn wir jetzt noch in die Lage kommen $\varphi(nm)$ für teilerfremde n und m zu berechnen, dann können wir $\varphi(n)$ für jede beliebige Zahlen bestimmen.

SATZ 5.10. *Für teilerfremde Zahlen n und m gilt*

$$\varphi(nm) = \varphi(n)\varphi(m).$$

BEWEIS: Wir zeigen, dass es eine eineindeutige Beziehung zwischen den Erzeugern von $\mathbb{Z}/nm\mathbb{Z}$ und Paaren von Erzeugern von $\mathbb{Z}/n\mathbb{Z}$ und $\mathbb{Z}/m\mathbb{Z}$ gibt. Wir betrachten dazu die Abbildung

$$f : \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z} \to \mathbb{Z}/mn\mathbb{Z} : (a, b) \mapsto ma + nb.$$

Da n und m teilerfremd sind, gibt es Zahlen s und t mit $sm + tn = 1$, in $\mathbb{Z}/nm\mathbb{Z}$ können wir sogar annehmen, dass $0 \leq s < n$ und $0 \leq t < m$

gilt. Da auch die Addition mit dieser Abbildung verträglich ist, folgt, dass sie eine Bijektion ist, und dass die beiden Mengen nur verschiedene Darstellungen der gleichen Gruppe sind. Jeder Erzeuger von $\mathbb{Z}/nm\mathbb{Z}$ gehört zu einem Erzeuger von $\mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}$. Es genügt also zu zeigen, dass die Paare (a, b) von Erzeugern von $\mathbb{Z}/n\mathbb{Z}$ und $\mathbb{Z}/m\mathbb{Z}$ auch Erzeuger im kartesischen Produkt sind.

Die Elemente $k(a, b)$ mit $0 \leq k < nm - 1$ sind alle verschieden. Wären nämlich zwei gleich müsste es k und k' geben mit $(k - k')a = 0$ in $\mathbb{Z}/n\mathbb{Z}$ und $(k - k')b = 0$ in $\mathbb{Z}/m\mathbb{Z}$. Daraus folgt, weil a zu n teilerfremd sein muss, dass m ein Teiler von $k - k'$ sein muss. Analog muss auch n ein Teiler von $k - k'$ sein. Da n und m teilerfremd sind, ist mn ein Teiler von $k - k'$, also ist $k = k'$. Somit ist $f(a, b)$ ein Erzeuger von $\mathbb{Z}/nm\mathbb{Z}$. Der Satz ist damit bewiesen.

Daraus lässt sich jetzt eine Formel für $\varphi(n)$ ableiten. Hat n die Primfaktorzerlegung

$$n = \prod_{i=0}^{k} p_i^{\alpha_i},$$

dann berechnet sich $\varphi(n)$ mit der Formel

$$\varphi(n) = \prod_{i=0}^{k} (p_i - 1) p_i^{\alpha_i - 1}.$$

Allgemeiner Satz von Fermat. Mit diesen Vorbereitungen können wir jetzt den kleine Satz von Fermat verallgemeinern:

Satz 5.11 (Fermat). *Für alle zu n teilerfremden a gilt $a^{\varphi(n)} = 1$ in $\mathbb{Z}/n\mathbb{Z}$.*

Beweis: Falls a teilerfremd ist zu n, ist a ein Erzeuger von $\mathbb{Z}/n\mathbb{Z}$, also ist

$$\{ka | (k, n) = 1\}$$

gerade die Menge aller Erzeuger. Wir berechnen das Produkt all dieser Elemente, und finden

$$\prod_{x \in \{ka | (k,n)=1\}} x = a^{\varphi(n)} \prod_{x \in \{k | (k,n)=1\}} x.$$

Die Menge der Erzeuger ist aber gerade die Indexmenge des rechten Produktes, es gilt also ausserdem

$$a^{\varphi(n)} \prod_{x\in\{k|(k,n)=1\}} x = \prod_{x\in\{k|(k,n)=1\}} x.$$

Und da alle k zu n teilerfremd sind, kann man das Produkt in $\mathbb{Z}/n\mathbb{Z}$ kürzen, und findet:

$$a^{\varphi(n)} = 1 \in \mathbb{Z}/n\mathbb{Z}.$$

Damit ist der kleine Satz von Fermat für a teilerfremd zu n bewiesen.

Im allgemeinen Fall kann man $\mathbb{Z}/n\mathbb{Z}$ zerlegen und die Abbildungen

$$\pi_i : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/p_i^{\alpha_i}\mathbb{Z}$$

konstruieren, welche nur den Rest eines Elementes in $\mathbb{Z}/n\mathbb{Z}$ bei Division durch $p_i^{\alpha_i}$ ermitteln. Nun gilt $\pi_i(a^{\varphi(n)}) = \pi_i(a)^{\varphi(n)} = 1$ für alle Faktoren, also muss auch $a^{\varphi(n)} = 1$ sein.

Produkt zweier Primzahlen. Aus dem Satz folgt $\varphi(pq) = (p-1)(q-1)$ für zwei Primzahlen p und q. Dies ist die Situation des im nächsten Abschnitt beschriebenen RSA-Verfahrens

5.4.2. RSA. Mit Hilfe des Produktes zweier grosser Primzahlen kann ein öffentliches Schlüsselverfahren aufgebaut werden. Dazu bestimmt jeder Teilnehmer zwei grosse Primzahlen p und q, und berechnet deren Produkt $n = pq$. Zu einer zu $\varphi(n)$ teilerfremden[6] Zahl e, die oft für alle Teilnehmer einheitlich gewählt wird, zum Beispiel 17 oder 65537, kann man ein d finden, welches die Eigenschaft hat, dass

$$ed + s\varphi(n) = 1,$$

bis auf ein Vielfaches von $\varphi(n)$ lässt sich e also invertieren. Nach dem kleinen Satz von Fermat gilt

$$m^{\varphi(n)} = 1 \in \mathbb{Z}/n\mathbb{Z}.$$

Das Paar (n, e) ist der öffentliche Schlüssel, (n, d) ist der private Schlüssel.

[6] Will man e einheitlich wählen, muss $p-1$ und $q-1$ teilerfremd zu e sein. Man kann die Primzahlen p und q nicht ganz beliebig wählen, sondern muss darauf achten, dass $p-1$ und $q-1$ keine kleinen Primfraktoren haben.

Ver- und Entschlüsselung. Eine Mitteilung $m < n$ wird jetzt durch Bildung von m^e in $\mathbb{Z}/n\mathbb{Z}$ verschlüsselt. Für die Entschlüsselung verwendet man

$$(m^e)^d = m^{ed} = m^{ed+s\varphi(n)} = m \in \mathbb{Z}/n\mathbb{Z},$$

die Entschlüsselung erfolgt also mit der gleichen Operation wie die Verschlüsselung, nur mit einem anderen Exponenten. Dies ist das nach den Erfindern Rivest, Shamir und Adleman benannte RSA-Verfahren.

Beispiel. Wir führen das RSA Verfahren ein einem Beispiel durch. Zunächst sind zwei Primzahlen p und q zu wählen, wir verwenden $p = 1307$ und $q = 1319$, der RSA-Modulus ist dann $pq = 1723933$. Als Verschlüsselungsexponent verwenden wir $e = 3$, für die Entschlüsselung müssen wir ein Inverses d modulo $(p-1)(q-1) = 1721308$ finden. Dazu muss der erweiterte euklidische Algorithmus auf 3 und 1721308 angewendet werden:

q	u_1	u_2	u_3	v_1	v_2	v_3
	1	0	1721308	0	1	3
573769	0	1	3	1	−573769	1
3	1	−573769	1	−3	1721308	0

Die Kontrolle ergibt $1 \cdot 1721308 - 573769 \cdot 3 = 1$, also ist $d = 1147539$ die gesucht Inverse.

Nun verschlüsseln wir die Meldung $m = 1291$ mit e:

$$m^e = 216787 \in \mathbb{Z}/1723933\mathbb{Z}.$$

Die Entschlüsselung ist

$$m^d = 216787^{1147539} = 1291 \in \mathbb{Z}/1723933\mathbb{Z},$$

wie zu erwarten war.

5.5. Kryptographie mit elliptischen Kurven.

Das RSA-Verfahren hat den Nachteil, relativ aufwendige Operationen mit grossen ganzen Zahlen durchführen zu müssen. In der Diskussion des Advanced Encryption Standard (Rijndael) haben wir gesehen, wie Multiplikationen in einem Körper $\mathbb{F}_{2^q}$ vergleichsweise einfach mit q-elementigen Bitvektoren durchgeführt werden können. Leider lässt sich das RSA-Verfahren nicht in einen solchen Körper abbilden.

Für das Diffie-Hellman-Verfahren sind jedoch wesentlich weniger Voraussetzungen notwendig: Es wird eine Menge E benötigt, in der eine Multiplikation definiert ist. Daraus kann man dann die Potenzen eines Elementes $g \in E$ mit dem bekannten Verfahren des Quadrierens und Multiplizierens berechnen. Diffie-Hellman lässt sich sich in die Menge E übertragen, ein sicheres Verfahren ergibt sich genau dann, wenn in E die Bestimmung von x aus g^x schwierig ist. Im klassischen Diffie-Hellman-Verfahren ist $E = \mathbb{F}_p^*$. Elliptische Kurven liefern eine Klasse von Mengen E, für die geforderten Eigenschaften erfüllt sind.

5.5.1. Was sind ellipitische Kurven? Elliptische Kurven sind Lösungemengen einer Gleichung der Form

$$Y^2 + XY = X^3 + aX + b,$$

wobei X und Y Elemente eines Körpers $\mathbb{F}_{p^l}$ sind:

$$E_{a,b}(\mathbb{F}_{p^l}) = \{(X, Y) \in \mathbb{F}_{p^l} | Y^2 + YX = X^3 + aX + b\}.$$

A priori ist nicht einmal bekannt, ob diese Gleichung überhaupt eine Lösung hat, umso überraschender ist, dass sich auf der Menge sogar eine gute Multiplikation definieren lässt.

Gruppenoperation in einem Spezialfall. Sofern $p \neq 2$ ist, könnte man durch quadratisches Ergänzen die Variable Y durch $Y + \frac{1}{2}X$ ersetzen und so den gemischten Term auf der linken Seite zum Verschwinden bringen. Ebenso könnte man durch eine Transformation in X eine neue Gleichung

$$u^2 = v^3 + Av + B$$

erhalten. Diese Form hat den Vorteil, dass die Lösungsmenge bezüglich Spiegelung an der x-Achse symmetrisch ist (Abbildung 5.1.).

Es ist naheliegend, die Gruppenoperation auf der Lösungsmenge so zu definieren, dass die Bildung der Inversen mit der Spiegelung an der x-Achse zusammenfällt.

Die Verknüpfung sollte ebenfalls geometrisch interpretiert werden können. Eine kubische Kurve hat die Eigenschaft, dass auf jeder Geraden durch zwei Punkte der Kurve ein weiterer Punkt der Kurve liegt. Sind g_1 und g_2 bereits Elemente der Kurve, und ist g_3 der genannte dritte Punkt,

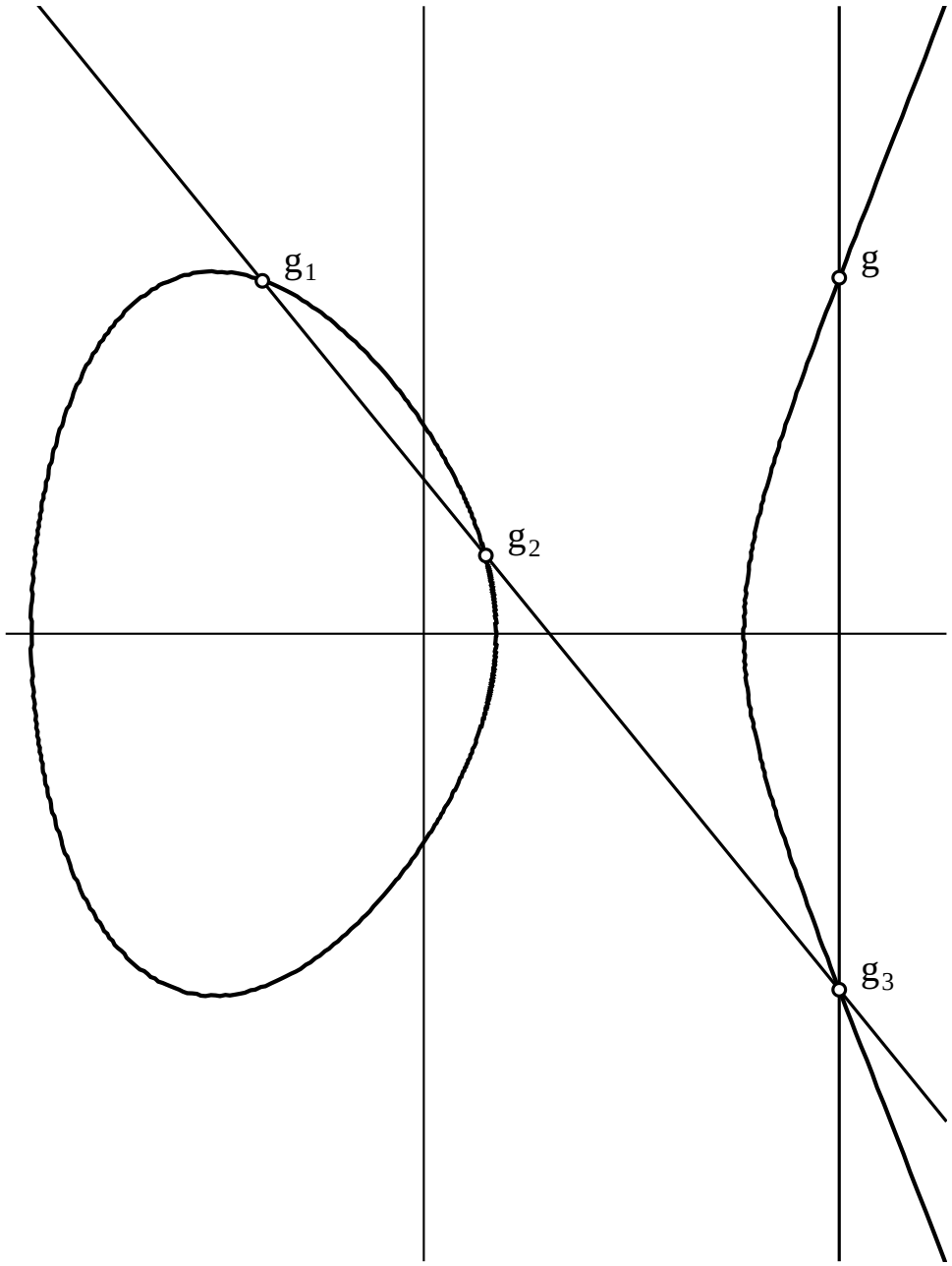

Abbildung 5.1. Gruppenoperation: $g_1 g_2 g_3 = 1$ und $g = g_3^{-1} = g_1 g_2$.

dann definieren wir die Verknüpfung so, dass $g_1 g_2 g_3 = 1$. Geometrisch konstruiert man die Verknüpfung also so, dass man den die Gerade durch g_1 und g_2 mit der Kurve schneidet, um g_3 zu erhalten, und diesen Punkt an der x-Achse spiegelt, also $g_1 g_2 = g_3^{-1}$.

Leider lässt sich diese Vereinfachung für $p = 2$ nicht durchführen, weil in $\mathbb{F}_{2^l}$ die Zahl 2 nicht invertierbar ist, also das quadratische Ergänzen

(bei dem durch 2 geteilt werden muss) nicht durchführbar ist. Daher muss die korrekte Definition der Gruppenoperation von der ursprünglich genannten Gleichung ausgehen.

Gruppenoperation im Allgemeinen Fall. Die Gruppenoperation der elliptischen Kurve muss für zwei Lösungen (X_1, Y_1), $(X_2, Y_2) \in E_{a,b}$ eine weiteres Paar $(X_3, Y_3) \in E_{a,b}$ produzieren. Ausserdem muss die Operation assoziativ sein, weil nur so die Definition g^x eindeutig ist[7].

Die Operation lässt sich über $\mathbb{R}$ geometrisch wie folgt veranschaulichen. Die Menge $E_{a,b}(\mathbb{R})$ ist die Nullstellenmenge einer Gleichung dritten Grades mit reellen Koeffizienten. Die Punkte $g_1 = (X_1, Y_1)$ und $g_2 = (X_2, Y_2)$ erfüllen die Gleichung. Die Gerade durch die beiden Punkte lässt sich wie folgt beschreiben:

$$(X, Y) = (X_1, Y_1) + t(X_2 - X_1, Y_2 - Y_1), t \in \mathbb{R}.$$

Durch Einsetzen in der Gleichung erhält man eine kubische Gleichung mit den Lösungen $t = 0$ und $t = 1$, folglich gibt es noch eine weitere Lösung $t = t_3$, zu der der Punkt

$$g_1 g_2 = (X_3, Y_3) = (X_1 + t_3(X_2 - X_1), Y_1 + t_3(Y_2 - Y_1)) \in E_{a,b}(\mathbb{R})$$

gehört, ein Kandidat für den gesuchten dritten Punkt. Nun wäre diese Lösung aber nicht symmetrisch, wir verlangen daher, dass $g_1 g_2 g_3 = 1$ ist, der dritte Punkt ist also das Inverse von $g_1 g_2$.

Das geometrische Bild zeigt unmittelbar, dass das so konstruierte Produkt kommutativ ist. Die Assoziativität ist dagegen schwieriger, sie besagt nämlich dass sich die Geraden durch $g_1 g_2$ und g_3 einerseits und durch g_1 und $g_2 g_3$ andererseits in einem Punkt auf der Kurve schneiden müssen.

In einem beliebigen Körper können wir die Elemente der Geraden durch g_1 und g_2 immer noch durch

$$y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

[7] Im Prinzip würde es natürlich ausreichen, dass die Assoziativität in der Menge von Punkten gilt, die sich als beliebige Produkte von g schreiben lassen. Leider ist diese Bedinung viel schwieriger nachzuprüfen als die Assoziativität für die ganze Gruppe. Es ist aber auch nicht nötig, denn die Konstruktion zeigt ja, dass für beliebig a und b eine "gute" Gruppenoperation definiert ist.

beschreiben. Wir setzen das in die Gleichung der elliptischen Kurve ein

$$y^2 + xy - x^3 - ax - b = 0.$$

Das so ermittelte Polynom in x muss sowohl durch $x - x_1$ also auch durch $x - x_2$ teilbar sein, es verbleibt ein linearer Term. Schreiben wir $\lambda = (y_2 - y_1)/(x_2 - x_1)$, finden wir mit einem Computeralgebra-System sofort

$$\begin{aligned}
x_3 &= \frac{-(x_2^3 - x_2^2 x_1 - x_2 x_1^2 + x_1^3) + (y_2 - y_1)^2 + x_1 y_1 - x_2 y_1 - x_1 y_2 + x_2 y_2}{(x_2 - x_1)^2} \\
&= \frac{-x_2^3 + x_2^2 x_1 + x_2 x_1^2 - x_1^3 + (y_2 - y_1)^2 + (x_1 - x_2)(y_1 - y_2)}{(x_2 - x_1)^2} \\
&= \frac{-x_2^2(x_2 - x_1) + (x_2 - x_1)x_1^2 + (y_2 - y_1)^2 + (x_2 - x_1)(y_2 - y_1)}{(x_2 - x_1)^2} \\
&= \frac{-(x_2^2 - x_1^2)(x_2 - x_1) + (y_2 - y_1)^2 + (x_2 - x_1)(y_2 - y_1)}{(x_2 - x_1)^2} \\
&= \frac{-(x_2 + x_1)(x_2 - x_1)^2 + (y_2 - y_1)^2 + (x_2 - x_1)(y_2 - y_1)}{(x_2 - x_1)^2} \\
&= -(x_2 + x_1) + \lambda^2 + \lambda
\end{aligned}$$

Daraus kann man auch y_3 bestimmen:

$$y_3 = y_1 + \lambda(-2x_1 - x_2 + \lambda^2 + \lambda)$$

Diese Ableitung funktioniert jedoch nicht, wenn $x_2 = x_1$ ist.

Betrachten wir zunächst den Fall $y_2 \neq y_1$: In diesem Fall ist die Gerade $x = x_1 = x_2$, also wird die Gleichung zu einer quadratischen Gleichung für y

$$y^2 - x_1 y = x_1^3 + ax_1 + b,$$

die aber keine weiteren Lösungen als y_1 und y_2 haben kann. Daher führen wir ein zusätzliches Element $e = (0, 0)$ ein, welches zwar keine Lösung der Gleichung ist, aber in $E_{a,b}$ die Rolle des neutralen Elementes übernehmen soll. In diesem Falle sind also die beiden Elemente invers zueinander. Ist

also $g = (x, y)$ eine Lösung, so finden wird die Inverse $g^{-1} = (x, y')$ dazu, wir setzen $y' = y + t$ und finden

$$\begin{aligned} y'^2 + xy' &= x^3 + ax + b \\ (y+t)^2 + x(y+t) &= x^3 + ax + b \\ y^2 + 2yt + t^2 + xy + xt &= x^3 + ax + b \\ 2yt + t^2 + xt &= 0 \\ (2y + t + x)t &= 0 \end{aligned}$$

Die letzte Gleichung hat zwei Lösungen: $t = 0$ und $t = -x - 2y$. Der ersten Lösung entspricht der Punkt g, die zweite Lösung führt auf $g^{-1} = (x, -x - y)$. Zur Kontrolle setzen wir dies in die Gleichung ein:

$$(x+y)^2 - x(x+y) = x^2 + 2xy + y^2 - x^2 - xy = y^2 + xy = x^2 + ax + b.$$

Damit haben wir eine Formel zur Berechnung der Inversen gefunden.

Wenn aber $g_1 = g_2$ gilt, könnte λ immerhin mit Hilfe der Ableitung berechnet werden, doch würden wird damit Methoden verwenden, die nur in $\mathbb{R}$ oder $\mathbb{C}$ anwendbar sind. Wir sind aber primär an einer Methode in $\mathbb{F}_{p^l}$ interessiert. Wir suchen jetzt g_1 so, dass $g_1^2 g_2 = e$. Die Gerade durch die Punkte g_1 und g_2 hat die Gleichung

$$y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) = y_1 + \lambda(x - x_1).$$

Setzt man dies in die Gleichung der elliptischen Kurve ein, ergibt sich eine kubische Gleichung für x, die bei x_1 eine doppelte Nullstelle haben muss, und bei x_2 eine einfache. Insbesondere darf bei Division $(x - x_1)^2$ kein Rest bleiben, also müssen die Koeffizienten des linearen Polynoms, welches als Rest bleibt, verschwinden. Mathematica findet die beiden Gleichungen

$$\begin{aligned} 2x_1^3 + \lambda x_1^2 - 2\lambda x_1 y_1 + y_1^2 - b &= 0 \\ \lambda(x_1^2 - 2x_1 y_1) + 2x_1^3 + y_1^2 - b &= 0 \\ 3x_1^2 + \lambda x_1 + y_1 - 2\lambda y_1 + a &= 0 \\ 3x_1^2 + y_1 + a + \lambda(x_1 - 2y_1) &= 0 \end{aligned}$$

Aus der letzten Gleichung leiten wir ab:

$$\lambda = \frac{2x_1^3 + y_1^2 - b}{x_1^2 - 2x_1 y_1}$$

Der Quotient kann mit Mathematica berechnet werden, und hat genau eine Nullstelle, nämlich x_2:

$$x_2 = -\lambda + \lambda^2 - 2x_1.$$

Damit haben wir x_2 berechnet, und können also auch y_2 berechnen: $y_2 = y_1 + \lambda(x_2 - x_1)$.

Wir fassen diese Resultate in einem Satz zusammen:

Satz 5.12 (Gruppenstruktur auf einer kubischen Kurve). *Die Menge*

$$E_{a,b}(K) = \{(X, Y) \in K^2 | Y^2 - XY = X^3 + aX + b\} \cup \{(0, 0)\}$$

erhält eine Gruppenstruktur wie folgt:

a) *Die Inverse eines Elementes $g = (x, y)$ ist $g^{-1} = (x, -x - y)$.*
b) *Das Produkt der Elemente $g_1 = (x_1, y_1)$ und $g_2 = (x_2, y_2)$ wird wie folgt berechnet:*
   (i) *Falls $x_1 \neq x_2$ ist $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ definiert und das Produkt ist*

$$x_3 = -(x_1 + x_2) + \lambda^2 + \lambda$$
$$y_3 = y_1 + \lambda(-2x_1 - x_2 + \lambda^2 + \lambda)$$

   (ii) *Falls $x_1 = x_2$ und $y_1 \neq y_2$ ist, sind die Elemente g_1 und g_2 zueinander invers, das Produkt ist $e = (0, 0)$.*
   (iii) *Falls $g_1 = g_2$ verwendet man die folgende Verdoppelungsregel. Mit*

$$\lambda = \frac{2x_1^3 + y_1^2 - b}{x_1^2 - 2x_1 y_1}$$

wird der Punkt $g_2 = g_1^2$

$$x_2 = -\lambda + \lambda^2 - 2x_1$$
$$y_2 = -x_1 - y_1 - \lambda(x_2 - x_1)$$

Für unsere Zwecke ist nur $K = \mathbb{F}_{2^l}$ notwendig, womit sich die Formeln stark vereinfachen lassen:

Satz 5.13 (Elliptische Kurven über $\mathbb{F}_{2^l}$). *Die Menge*

$$E_{a,b}() = \{(X,Y) \in \mathbb{F}_{2^l}^2 | Y^2 - XY = X^3 + aX + b\} \cup \{(0,0)\}$$

erhält eine Gruppenstruktur wie folgt:

a) *Die Inverse eines Elementes $g = (x,y)$ ist $g^{-1} = (x, x+y)$.*
b) *Das Produkt der Elemente $g_1 = (x_1, y_1)$ und $g_2 = (x_2, y_2)$ wird wie folgt berechnet:*
   (i) *Falls $x_1 \neq x_2$ ist $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ definiert und das Produkt ist*

$$x_3 = x_1 + x_2 + \lambda^2 + \lambda$$
$$y_3 = y_1 + \lambda(x_2 + \lambda^2 + \lambda)$$

   (ii) *Falls $x_1 = x_2$ und $y_1 \neq y_2$ ist, sind die Elemente g_1 und g_2 zueinander invers, das Produkt ist $e = (0,0)$.*
   (iii) *Falls $g_1 = g_2$ verwendet man die folgende Verdoppelungsregel. Mit*

$$\lambda = \frac{y_1^2 + b}{x_1^2}$$

wird der Punkt $g_2 = g_1^2$

$$x_2 = \lambda + \lambda^2$$
$$y_2 = x_1 + y_1 + \lambda(x_2 + x_1)$$

Implementation. In den praktischen Implementationen verwendet man für die Konstruktion der Körper $\mathbb{F}_{2^l}$ Polynome, die nur wenige nichtverschwindende Koeffizienten haben, die zudem weit auseinander liegen. Die Polynom-Division kann in diesem Fall effizienter realisiert werden.

5.5.2. Diffie-Hellman mit elliptischen Kurven. Das Diffie-Hellman-Verfahren verlangt gar nicht, dass die Zahlen addiert werden können müssen, einzige eine effiziente Operation $x \mapsto g^x$, die nur sehr schwierig umzukehren ist, wird benötigt. Nun ist aber in der elliptischen Kurve

eine Multiplikation definiert, und das Verfahren des Quadrierens und Multiplizieren liefert einen effizienten Algorithmus um g^x zu berechnen. Nun muss nur noch durch geeignete Wahl von $g \in E$ sichergestellt werden, dass zunächst einmal die Menge $\{g^k \in E | k \in K\}$ sehr gross ist, aber auch das Problem des diskreten Logarithmus in der elliptischen Kurve schwierig ist:

Problem 5.14 (Diskreter Logarithmus in einer elliptischen Kurve). *Zu geeignetem g in der elliptischen Kurve E ist es praktisch unmöglich, aus $g^x \in E$ den Exponenten x zu bestimmen.*

Dieses Verfahren wird gemäss RFC 2409 in IPSec in den Oakley-Gruppen 3 und 4 für die Aushandlung eines Sitzungsschlüssels verwendet. Wie früher gezeigt lassen sich Elemente als $\mathbb{F}_{2^l}$ als Bitvektoren schreiben. Die beiden Oakley-Gruppen verwenden verschiedene Grundkörper mit verschiedenen Ordnungen. Oakley-Gruppe 3:

$$\begin{aligned} K &= \mathbb{F}_{2^{155}} = \mathbb{F}_2[u]/(u^{155} + u^{62} + 1)\mathbb{F}_2[u] \\ g &= \texttt{0x7bb} \\ a &= \texttt{0x0} \\ b &= \texttt{0x7338f} \end{aligned}$$

Oakley-Gruppe 4:

$$\begin{aligned} K &= \mathbb{F}_{2^{185}} = \mathbb{F}_2[u]/(u^{185} + u^{69} + 1)\mathbb{F}_2[u] \\ g &= \texttt{0x18} \\ a &= \texttt{0x0} \\ b &= \texttt{0x1ee9} \end{aligned}$$

5.6. Anwendungen.

Mit RSA und seinen Verwandten als Bausteine lassen sich vielfältige kryptographische Anwendungen zusammenbauen. Als Beispiele seien in diesem Abschnitt nur die elektronische Unterschrift und die blinde Unterschrift genannt.

5.6.1. Elektronische Unterschrift mit RSA. Eine Unterschrift ist eine zu einem Dokument gehörige Information, die nur der Autor des

Unterschrift erstellen kann, aber von jederman überprüft werden kann. RSA ermöglicht sofort elektronische Unterschriften zu leisten: Die Meldung m wird von A unterschrieben, indem er sie mit seinem privaten Schlüssel verschlüsselt, also $s = m^d$ (in $\mathbb{Z}/pq\mathbb{Z}$) berechnet. Jederman hat Zugang zu e, und kann damit aus der Unterschrift s wieder $s^e = m^{de} = m$ berechnen, und damit die Unterschrift überprüfen.

In der Praxis ist dieses Verfahren nicht unmittelbar anwendbar: Dokumente sind meist grösser, als was mit RSA verschlüsselt werden kann, und selbst wenn ein kurzes Dokument vorliegt, hängt die Überprüfung der Unterschrift davon ab, ob man s^e als gültig erkennen kann. Die praktische Durchführung unterschreibt daher nicht die Meldung direkt, sondern leitet aus dem Dokument mit Hilfe einer Hashfunktion h einen Hashwert $h(m)$ ab. Dieser ist klein genug, mit RSA verschlüsselt zu werden, die Unterschrift ist jetzt also $h(m)^d$. Die Unterschrift wird überprüft, indem man zunächst $h(m)$ berechnet, und dann kontrolliert, ob $s^e = (h(m)^d)^e = h(m)^{de} = h(m)$ damit übereinstimmt.

Die so konstruierte Unterschrift hat den zusätzlichen Vorteil, dass man die Unterschrift nicht überprüfen muss, um den Inhalt der Mitteilung lesen zu können.

5.6.2. Blinde Unterschrift. Mit RSA kann auch eine blinde Unterschrift geleistet werden. Möchte B, dass A ein Dokument m unterschreibt, ohne dessen Inhalt je zu Gesicht bekommen zu haben, kann sie wie folgt vorgehen. Zunächst erzeugt Sie eine Zufallszahl v, die in $\mathbb{Z}/pq\mathbb{Z}$ invertierbar ist, und multipliziert die Meldung m mit v^e (in $\mathbb{Z}/pq\mathbb{Z}$). Nun verlangt A von B, das Produkt mv^e zu verschlüsseln, was nur A als Besitzerin des privaten Schlüssels d kann. Sie erhält $(mv^e)^d = m^d v^{ed} = m^d v$. Nun kann B ohne Probleme mit dem nur ihr bekannten v^{-1} multiplizieren, um die Unterschrift m^d zu erhalten.

5.6.3. Elektronische Unterschrift mit ElGamal. Zunächst definiert A seinen öffentlichen Schlüssel: eine Primzahl p, ein Element $g \in \mathbb{F}_p$ und $y = g^x \in \mathbb{F}_p$, wobei $x < p$ der private Schlüssel ist. Um die Mitteilung M zu unterschreiben wählt er dann ein zufälliges k teilerfremd zu $p - 1$ und berechnet einerseits

$$a = g^k \in \mathbb{F}_p$$

und andererseits b so, dass

$$\begin{aligned} M &\equiv xa + kb \mod p-1 \\ M - xa &= kb + s(p-1). \end{aligned}$$

Das Paar (a, b) ist die Unterschrift.

Die Zahl b kann mit Hilfe des euklidischen Algorithmus gefunden werden: weil $p-1$ und k teilerfremd sind, gibt es Zahlen b' und t mit $1 = kb' + t(p-1)$. Durch Multiplikation mit $M - xa$ erhält man

$$\begin{aligned} M - xa &= kb'(M - xa) + (p-1)s(M - xa) \\ M &\equiv xa + kb \mod p-1 \end{aligned}$$

mit $b = b'(M - xa)$.

Die Unterschrift wird verifiziert, indem man in $\mathbb{F}_p$

$$y^a a^b = g^{xa} g^{kb} = g^{xa+kb} = g^M$$

nachrechnet.

5.6.4. Schnorr-Authentisierung. Zunächst werden zwei Primzahlen p und q vorbereitet, wobei q in Primfaktor von $p-1$ sein muss. Wähle a so, dass $a^q = 1 \in \mathbb{F}_p$. Alle diese Werte sind öffentlich.

Der private Schlüssel ist eine zufällige Zahl $s < q$. Der zugehörige öffentliche Schlüssel ist $v = a^{-s} \in \mathbb{F}_p$. Ziel des Verfahrens ist, dass Peggy, die Besitzerin des privaten Schlüssels ihre Kenntnis Victor, der nur den öffentlichen Schlüssel kennt, zu beweisen, ohne etwas über den Schlüssel zu verraten.

1. Peggy wählt eine Zufallszahl $r < q$ und übermittelt $x = a^r \in \mathbb{F}_p$ an Victor.
2. (Challenge) Victor übermittelt eine Zufallszahl e mit $0 < e < 2^t - 1$, in den Anwendungen ist t typischerweise etwa 72.
3. Peggy berechnet $y = r + se \mod q$ und übermittelt y an Victor.
4. Victor berechnet $a^y v^e \in \mathbb{F}_p$

Victor findet

$$a^y v^e = a^{r+se} a^{-se} = a^r = x \in \mathbb{F}_p,$$

wenn also das Resultat von Victor's Rechnung mit der von Peggy im ersten Schritt gemeldeten Grösse x übereinstimmt, kann Victor annehmen, dass sein Kommunikationspartner tatsächlich Peggy ist, oder mindestens das Geheimnis s kennt.

5.6.5. X.509-Zertifikate. X.509-Zertifikate bilden eine Infrastruktur für den Aufbau von Vertrauen. In der Diskussion des Diffie-Hellman-Verfahrens wurde am Beispiel von SRP gezeigt, wie der mögliche Man-in-the-Middle-Angriff durch zusätzliche Authentisierung des Servers abgewendet werden kann. Für Internet-Anwendungen ist dies jedoch nur begrenzt tauglich, da der Benutzer vorgängig in der Benutzerdatenbank registriert werden musste.

Ein Internet-taugliches System muss ermöglichen, dass man mit einem Server Verbindung aufnehmen und sich darauf verlassen kann, tatsächlich mit dem erwarteten Server zu sprechen, ohne dass vorher irgend ein Austausch notwendig war. Möglich wird dies, wenn der Server eine Bestätigung einer dritten, beiden Teilnehmern vertrauenswürdigen Stelle vorlegen kann, die dem Server bescheinigt, tatsächlich der Eigentümer des angewählten URL zu sein.

Eine solche Bescheinigung ist ein Zertifikate, es bezeugt durch eine elektronische Unterschrift, dass ein gewisser öffentlicher Schlüssel zu einer Identität gehört.

Ein Vergleich. Nicht elektronische Formen von Zertifikaten sind im modernen Leben nicht mehr wegzudenken:

Pass. In einem Pass bestätigt ein Passbüro die Personalien des Besitzers des zum Passphoto (öffentlicher Schlüssel) passenden Gesichtes (privater Schlüssel). Bei der Passkontrolle überprüft der Beamte, ob der private und und der öffentliche Schlüssel zusammenpassen, und ob der Pass von einer Stelle ausgestellt wurde, der er trauen kann. In seltenen Fällen fragt er auch ein Liste ab, die alle nicht mehr gültigen Pässe eines Passbüros auflistet.

SBB Abonnement. Auf ihren Abonnement-Karten bestätigt die SBB gewisse Beförderungsrechte des Besitzers des zum Photo (öffentlicher Schlüssel) passenden Gesichtes (privater Schlüssel). Der Kodukteur kontrolliert das Zusammenpassen von privatem und öffentlichem Schlüssel, die Gültigkeitsdauer des Abonnements und ob die vom Abonnement bestätigten Rechte für die aktuelle Beförderung

ausreichen.

Fahrausweis. Das Strassenverkehrsamt bestätigt in einem Fahrausweise die Berechtigung des Besitzers des zum Photo passenden Gesichtes, gewisse Kategorien von Motorfahrzeugen zu führen. Bei einer Ausweiskontrolle überprüft der Polizist das Zusammenpassen von privatem und öffentlichem Schlüssel, die Gültigkeit des Ausweises und ob die vom Ausweis bestätigten Berechtigungen zum Führen des aktuellen Fahrzeuges ausreichen.

Inhalt eines Zertifikates. Aus obigen Beispielen lässt sich ablesen, welche Art von Informationen in einem Zertifikat vorhanden sein müssen:

Identität des Herausgebers. Mit Hilfe dieser Beschreibung der Identität des Herausgebers und der Unterschrift (weiter unten) muss es möglich sein, die Echtheit des Zertifikates zu prüfen.

Identität des Eigentümers. Die Beschreibung des Eigentümers des Zeritifikates. Sie dient der einfacheren Identifikation des Zeritifikates, oder enthält einen Schlüssel, der in einem Anwendungsfall auf weitere Informationen zum Eigentümer hinweist. Verwendet man einen Fahrausweis als Identitätszertifikat für eine Banktransaktion, kontrolliert der Schalterbeamte die Gültigkeit und das Zusammenpassen von privatem und öffentlichem Schlüssel (Gesicht und Photo), und sucht dann mit Hilfe von Name und Adresse nach den Kundeninformationen.

Gültigkeit Beim Einsatz des Zertifikates muss überprüft werden können, ob es überhaupt noch gültig ist. Verlorene Zertifikate sollen nur während einer beschränkten Zeit eine Bedrohung darstellen können. Die Anwendung SBB-Abonnement zeigt deutlich, dass Zertifikate für ganz bestimmte, relativ kurze Zeitintervalle erstellt werden können, daher ist auch der Begin der Gültigkeit festzuhalten. Für die spätere Überprüfung einer elektronischen Unterschrift ist wesentlich, ob die Unterschrift zu dem Zeitpunkt, da sie geleistet wurde, überhaupt gültig war.

Seriennummer. Die Seriennummer erlaubt eine rasche Identifikation eines bestimmten Zertifikates innerhalb einer Liste aller Zertifikate, die der Herausgeber zu führen hat.

Öffentlicher Schlüssel. In den Anwendungsbeispielen ein Passphoto,

in der Annahme, dass sich physische Gesichter nur sehr schwer kopieren lassen. In X.509-Anwendungen ein RSA public key, also ein Paar (n, e).

Unterschrift. In den Beispielen erfolgt die Echtheitsbestätigung oft durch spezielle Formate (Karten mit Hologrammen), Stempel oder (mehrfache) Unterschriften.

In der Praxis können "Zertifikate" noch viele weitere Informationen enthalten. So kann zum Beispiel in einem Fahrausweis ein Attribut stehen, welches den Eigentümer verpflichtet, beim Autofahren eine Brille zu tragen.

Beispiel eines X.509 Zertifikates. Das folgende Bespiel soll den Aufbau eines X.509 Zertifikates klären. Es handelt sich um ein Zertifikat, welches von Thawte Consulting in Südafrika für den Webserver `www.tpl.ch` der Firma Transparent Paper Ltd. erstellt wurde.

```
Data:
    Version: 3 (0x2)
    Serial Number: 41623 (0xa297)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
        OU=Certification Services Division, CN=Thawte Server CA/
        Email=server-certs@thawte.com
    Validity
        Not Before: Nov 24 14:49:01 1999 GMT
        Not After : Dec  7 14:49:01 2000 GMT
    Subject: C=CH, ST=Zurich, L=Zurich, O=Transparent Paper Ltd.,
        CN=www.tpl.ch
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
        RSA Public Key: (1024 bit)
            Modulus (1024 bit):
                00:be:73:5d:0d:31:69:92:49:e7:f8:45:9c:a7:b8:
                f0:4a:38:a1:ce:15:c2:78:c6:6f:01:83:2d:43:c4:
                0f:7e:ed:dc:1a:89:8b:18:d4:93:24:43:8e:6d:18:
                e3:7d:2b:2a:ca:0a:05:32:b2:77:eb:e4:8f:e8:f5:
                17:e3:b2:56:23:b7:80:7b:2c:0c:bd:4c:45:5a:c7:
                cf:1a:30:68:96:e3:21:62:c1:c2:66:96:91:a1:aa:
                dc:00:d7:b3:f0:9d:76:1d:20:ad:2c:8c:95:9d:b1:
                e2:ea:c7:ed:a3:a1:a5:01:a2:9a:d7:c4:05:84:3f:
                da:a2:00:ef:97:6c:b5:c5:85
```

```
            Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Basic Constraints: critical
                CA:FALSE
    Signature Algorithm: md5WithRSAEncryption
        b8:64:b4:5f:e9:fb:c1:30:62:4a:0c:a5:e5:86:6e:13:66:c3:
        bf:3f:ce:63:40:0b:e1:53:93:e5:34:9f:bb:f4:93:cc:31:23:
        08:f9:28:7c:2d:f9:17:96:a5:b2:02:9c:1d:c6:b4:e4:5f:53:
        8b:bd:fa:d7:db:dc:ee:80:56:47:1c:28:3a:ac:ca:15:c3:8c:
        f0:e6:c3:dd:f5:fb:04:6a:05:5d:fa:27:4a:5b:e4:0e:12:6f:
        43:be:55:fe:06:73:61:50:e4:e1:01:60:ba:e2:82:96:34:1e:
        b8:6f:e0:da:84:5a:0f:e6:54:85:4a:54:e0:d8:c1:b1:04:f7:
        25:6b
```

Das Zertifikat besteht aus zwei Hauptteilen: den Daten (`Data`) und der Unterschrift (`Signature Algorithm:`). Die Unterschrift bestätigt den Datenblock als Ganzes.

Die Identität von Herausgeber und Eigentümer des Zertifikates sind als X.500 distinguished Names zu lesen. Dahinter steckt die Idee, dass jede Entität in einem globalen X.500-Verzeichnis über Zertifikate verfügen könnte. Leider enthalten X.500 distinguished names zusätzliche Informationen, die in der menschenlesbaren Form nicht erkenntlich sind (String-Typ), und die in die weitaus gebräuchlicheren LDAP-Verzeichnisse nicht abgebildet werden.

Die Unterschrüft verwendet RSA als Verschlüsselung, kann also nur mit einem geeigneten RSA public key überprüft werden, der im Zertifikat nicht enthalten ist. Er kann jedoch auf Grund der Beschreibung der Identität des Herausgebers im Feld `Issuer` ermittelt werden. Es muss ein Zertifikat geben, welches den in diesem Feld angegeben Namen sowohl Herausgeber wie auch Besitzer hat, in dem der öffentliche Schlüssel der Thawte Zertifizierungsstelle steht, die mit dem dazu passenden privaten Schlüssel unterschrieben ist. Man muss also auf irgend einem anderen Weg dazu kommen, dieses Ausgangszertifikat zu erhalten, damit man ihm trauen kann.

Die Seriennummer erlaubt, das Zeritifikat in einer periodisch veröffentlichten Liste noch nicht abgelaufener, aber trotzdem ungültiger Zertifikate zu finden. Diese sogenannte certificate revocation list CRL enthält

alle Zertifikate, die gestohlen oder verloren wurden, oder die aus anderen Gründen vorzeigt nicht mehr verwendet werden dürfen.

Ein Feld von Erweiterungen ermöglich, in dem Zertifikat zusätzliche Informationen festzuhalten, in diesem Beispiel wird verboten, dass der private Schlüssel zu diesem Zertifikat zum Ausstellen weiterer Zertifikate benutzt wird. Ausserdem wird festgehalten, dass es ausschliesslich für die Authentisierung von Webservern gedacht ist.

Ein einmal erstelltes X.509-Zertifikat kann nicht mehr geändert werden, es ist also nicht möglich, wie in einem Pass zusätzliche Eintragungen für neue Berechtigungen (zum Beispiel ein Visum, welches eine Einreise- oder Aufenthaltsberechtigung darstellt) vorzunehmen. In X.509-Anwendungen müsste dies durch zusätzliche Zertifikate geschehen, die dem Besitzer eines Identitätszertifikates zusätzliche Attribute attestieren (Attribut-Zertifikate).

Übungsaufgabe: Überlegen Sie sich, inwieweit Banknoten Zertifikate sind.

LITERATUR

1. Donald E. Knuth, *The Art of Computer Programming*, Seminumerical Algorithms, 2nd Edition, Addison Wesley, Reading, Massachusetts.
2. Peter Mongomery, *Modular Multiplication Without Trial Division*, Mathematics of Computation **170**, 519-521.
3. J. Daemen, V. Rijmen, *AES Proposal: Rijndael*.
4. Bruce Schneier, *Applied Cryptography*, 2nd Edition, John Wiley & Sons, Inc., New York.
5. Reinhard Wobst, *Abenteuer Kryptologie*, 2., überarbeitete Auflage, Addison Wesley, Bonn; Reading, Massachussetts.